Swascan

TINEXTA GROUP

LockBit 3.0 Leak

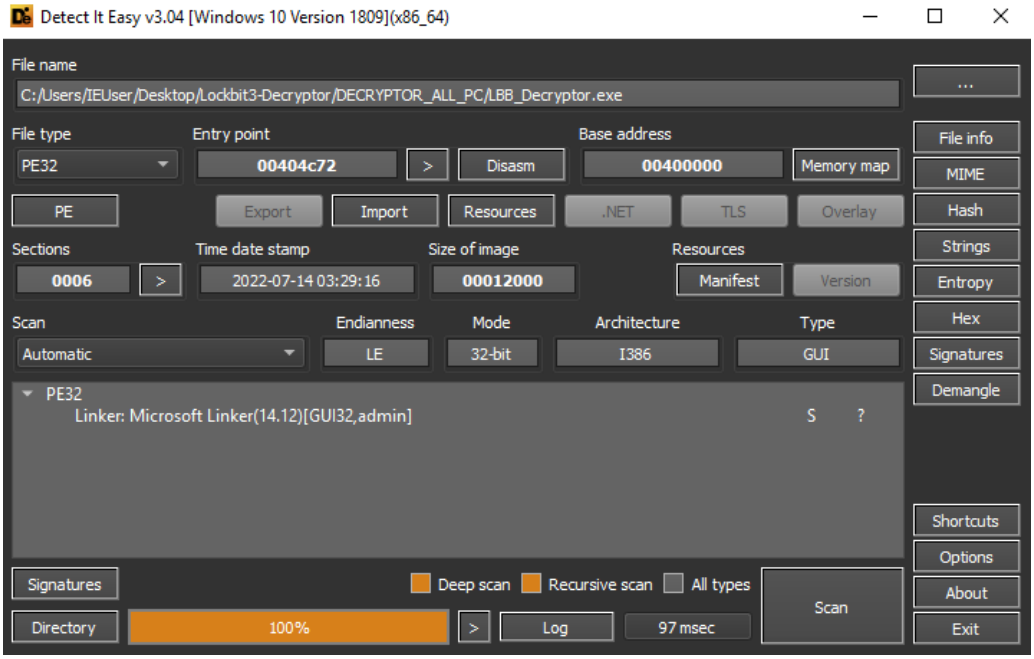
LockBit 3 Decryptors analysis

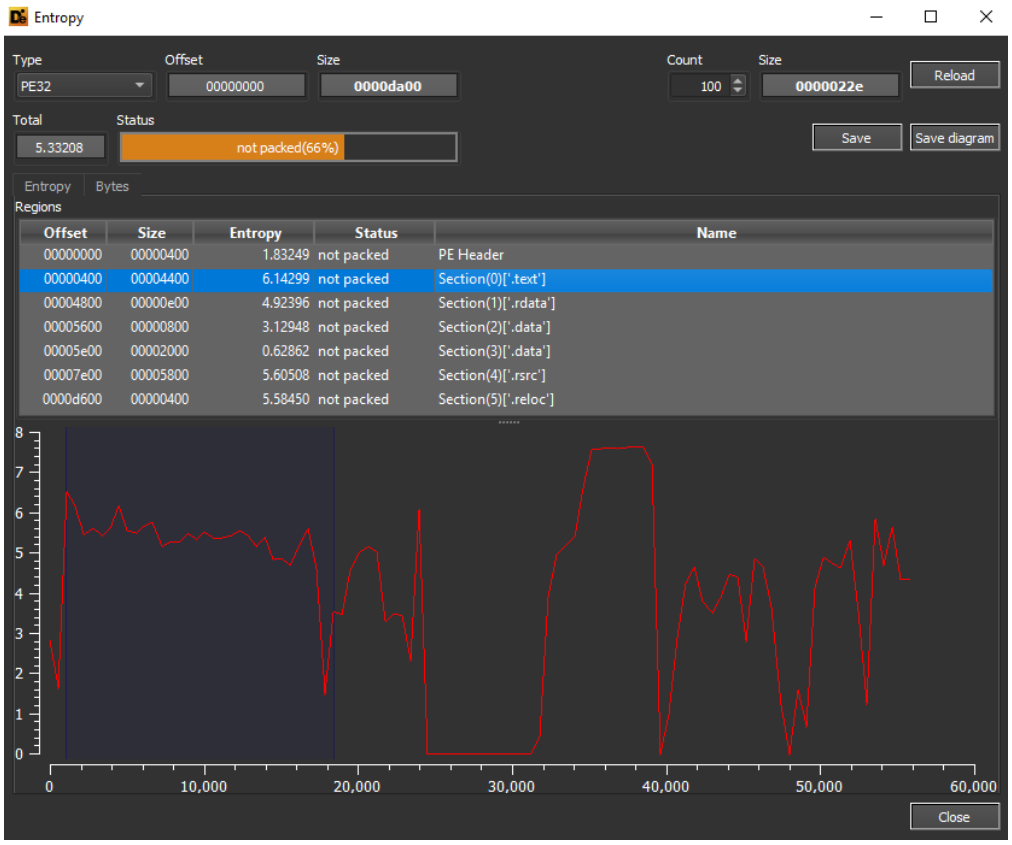
In this analysis we take in consideration the decryptor tools of LockBit 3.0 ransomware (Windows version) and LockBit (ESXi variant).

Generally, a decryptor tool permits to decrypt the files encrypted by a ransomware threat; this type of tool could be delivered directly by the ransomware gang, who developed the threat (after the payment of the requested ransom) or is developed ad-hoc from the global cybersecurity community to be distributed to help the victims to decrypt their files without the need to actually pay the ransom. Unfortunately, this is not always possible, however the cybersecurity community constantly work to develop new decryptors for the ransomware threats and the related variants. For this reason, it is not recommended to remove the encrypted files, because it is not excluded that, in a future, they can effectively be decrypted. Usually, the decryptors developed by the cybersecurity community take in consideration the ransomware threats (and the possible variants) which use an offline encryption key. In this case the analyzed decryptors have been downloaded from the Web.

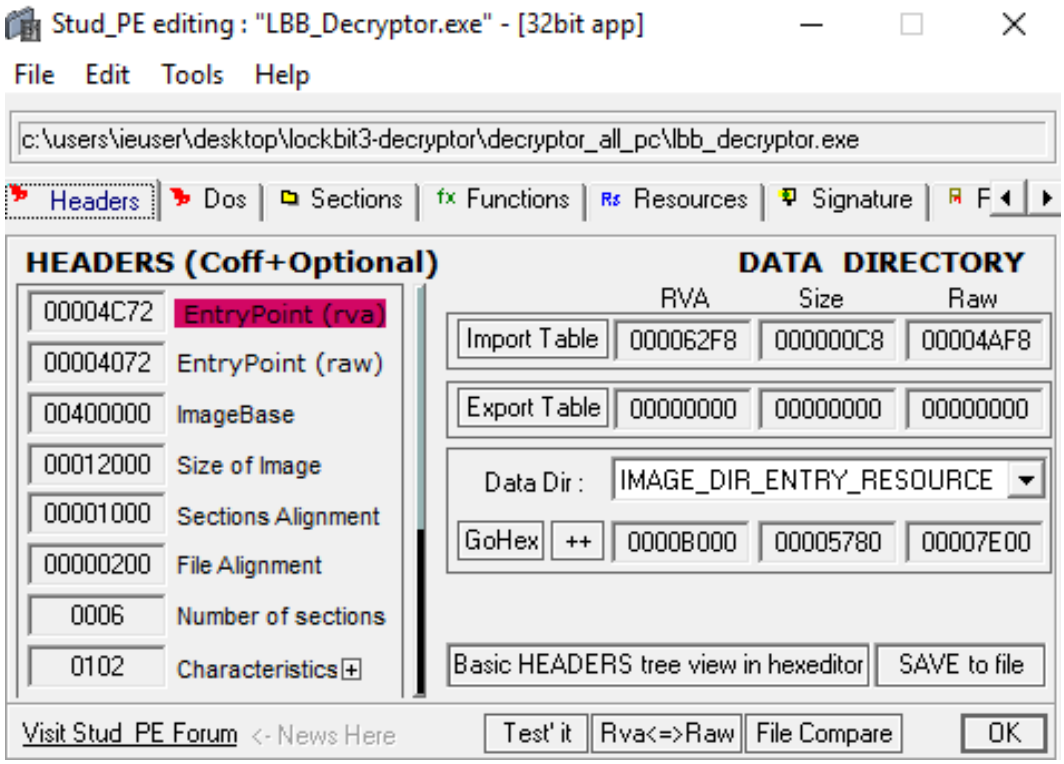
The executable **LBB_Decryptor.exe** doesn't have a high level of entropy, which would indicate a "packed" state, in the imports we can see that the mpr.dll library is used to execute network checking functions, SHLWAPI.dll to manage filesystem items, search the paths of files to decrypt, check the existence of the files, if some directories are empty, etc.

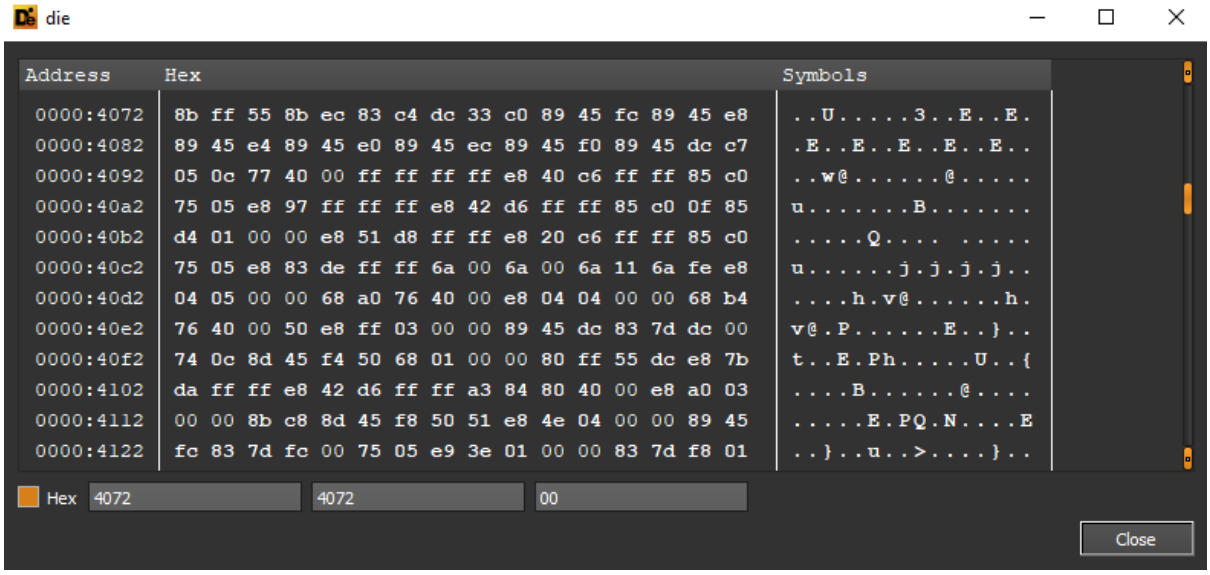
AdvAPI32.dll library is used to call functions to manage the ***MD5*** contained in the **README** file and manipulate (restore) registry keys related to the infection (Create, Delete, SetValue operations). The library SHELL32.dll, instead, is imported to execute the function **SHGetSpecialFolderPathW** to access to the settings of the file **NTUSER.dat**; there are also evidences associated to the creation and management of threads to execute decryption functions in a more efficient way.


Detect It Easy v3.04 [Windows 10 Version 1809](x86_64)
File name
C:/Users/IEUser/Desktop/Lockbit3-Decryptor/DECRYPTOR_ALL_PC/LBB_Decryptor.exe
File type
PE32
Entry point
00404c72
Disasm
Base address
00400000
Memory map
PE
Export
Import
Resources
.NET
TLS
Overlay
Sections
0006
Time date stamp
2022-07-14 03:29:16
Size of image
00012000
Resources
Manifest
Version
Scan
Automatic
Endianness
LE
Mode
32-bit
Architecture
I386
Type
GUI
PE32
Linker: Microsoft Linker(14.12)[GUI32,admin]
S
?
File info
MIME
Hash
Strings
Entropy
Hex
Signatures
Demangle
Shortcuts
Options
About
Exit
Signatures
Deep scan
Recursive scan
All types
Scan
Directory
100%
Log
97 msec

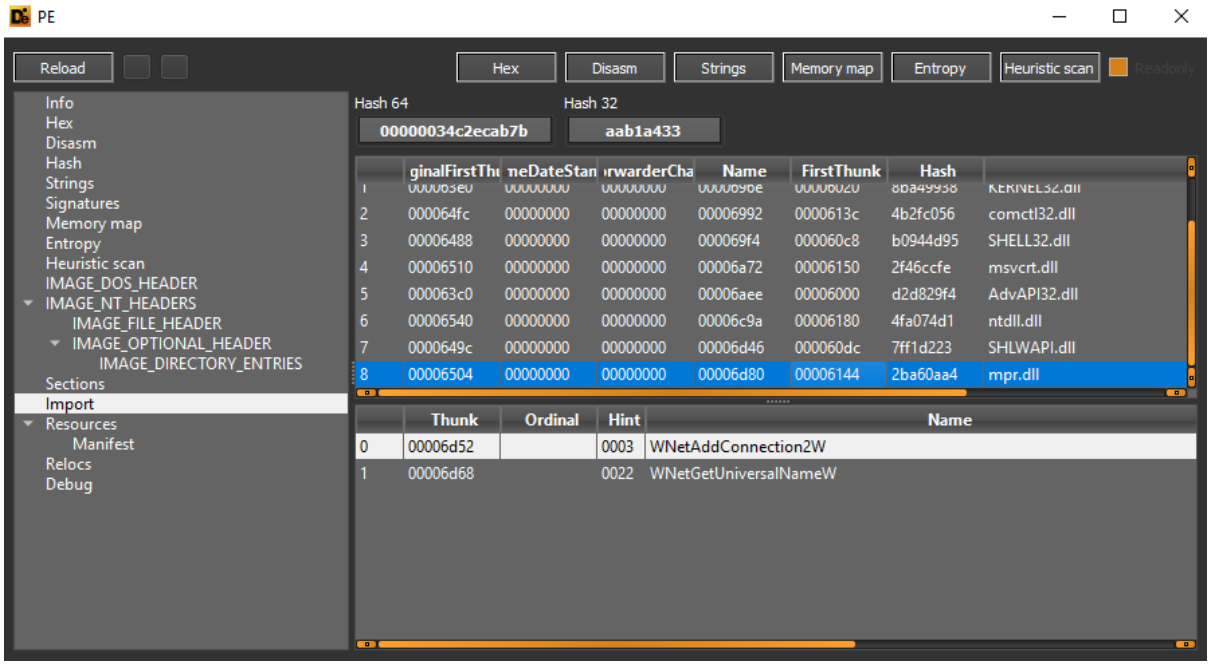

Entropy
Type
PE32
Offset
00000000
Size
0000da00
Count
100
Size
0000022e
Reload
Total
5.33208
Status
not packed(66%)
Save
Save diagram
Entropy
Bytes
Regions
Offset
Size
Entropy
Status
Name
00000000 00000400 1.83249 not packed PE Header
00000400 00004400 6.14299 not packed Section(0)['.text']
00004800 00000e00 4.92396 not packed Section(1)['.rdata']
00005600 00000800 3.12948 not packed Section(2)['.data']
00005e00 00002000 0.62862 not packed Section(3)['.data']
00007e00 00005800 5.60508 not packed Section(4)['.rsrc']
0000d600 00000400 5.58450 not packed Section(5)['.reloc']
8
7
6
5
4
3
2
1
0
0
10,000
20,000
30,000
40,000
50,000
60,000
Close

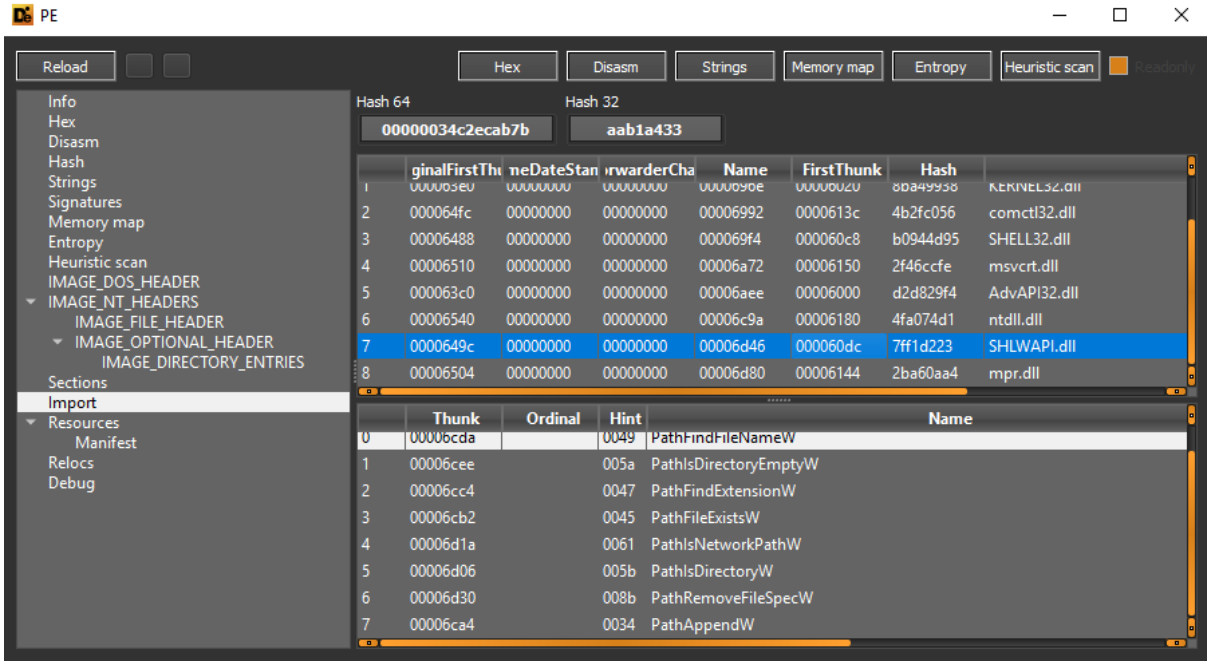
property	value
md5	EEE0D4094CD4E7CF318DA3E7E29E90E2
sha1	34ED29F3CD23B7662ADAF4D8B8AD6D68C5298AA6
sha256	E73083AC86F25E8DA3AA28547ED386F5336C57CFD191117DEA3451498FEB36F1
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes-text	M Z .. .. @ .. ..
file-size	55808 (bytes)
entropy	5.332
imphash	3E00A7DD9BB741EE6A4F22F50F51341A
signature	n/a
entry-point	8B FF 55 8B EC 83 C4 DC 33 C0 89 45 FC 89 45 E8 89 45 E4 89 45 E0 89 45 EC 89 45 F0 89 45 DC C7 05
file-version	n/a
description	n/a
file-type	**executable**
cpu	**32-bit**
subsystem	GUI
compiler-stamp	0x62CFEFFC (Thu Jul 14 03:29:16 2022)
debugger-stamp	0x62CFEFFC (Thu Jul 14 03:29:16 2022)
resources-stamp	0x00000000 (empty)
import-stamp	0x00000000 (empty)
exports-stamp	n/a
version-stamp	n/a
certificate-stamp	n/a

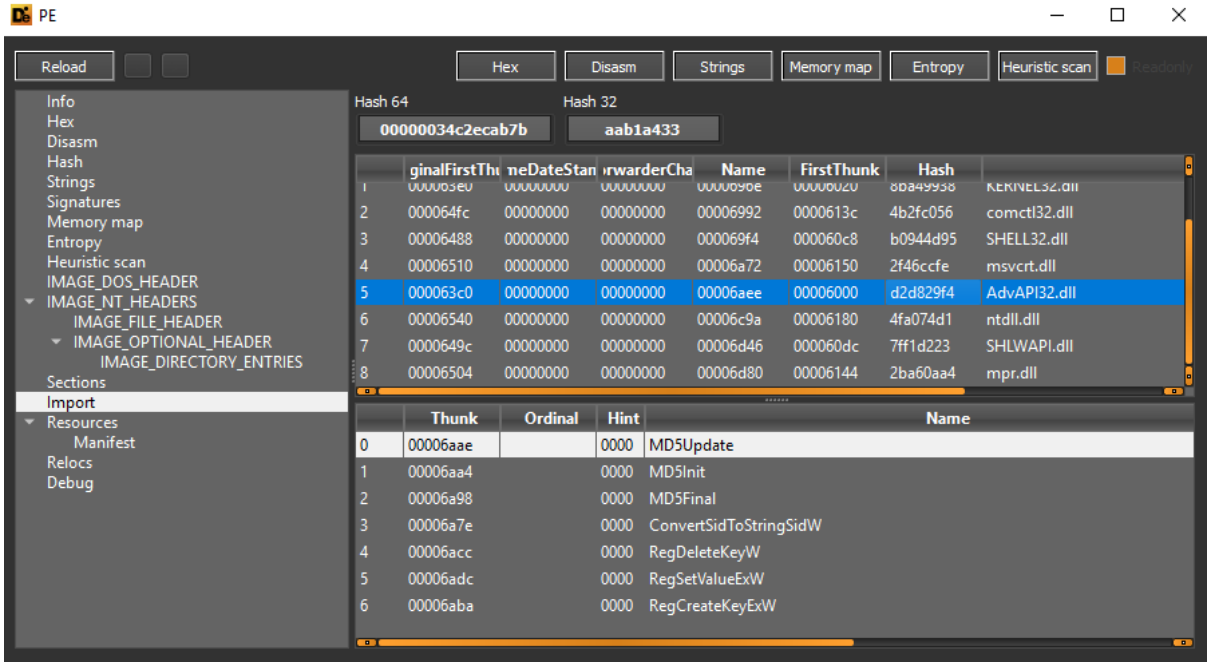


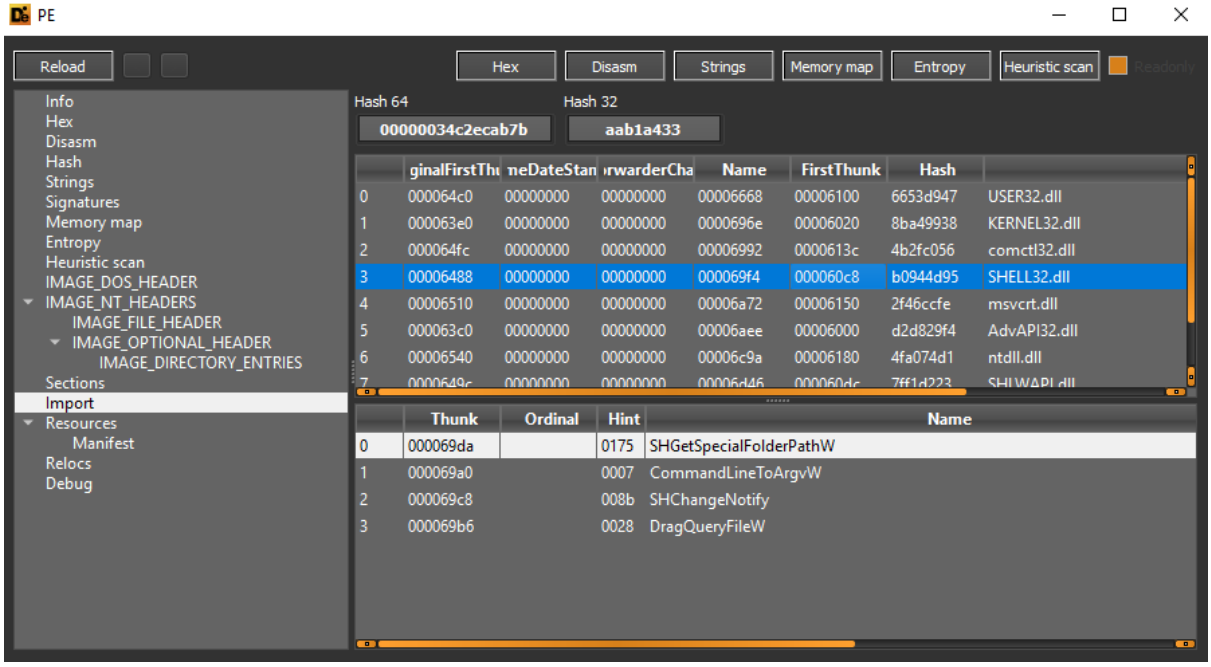


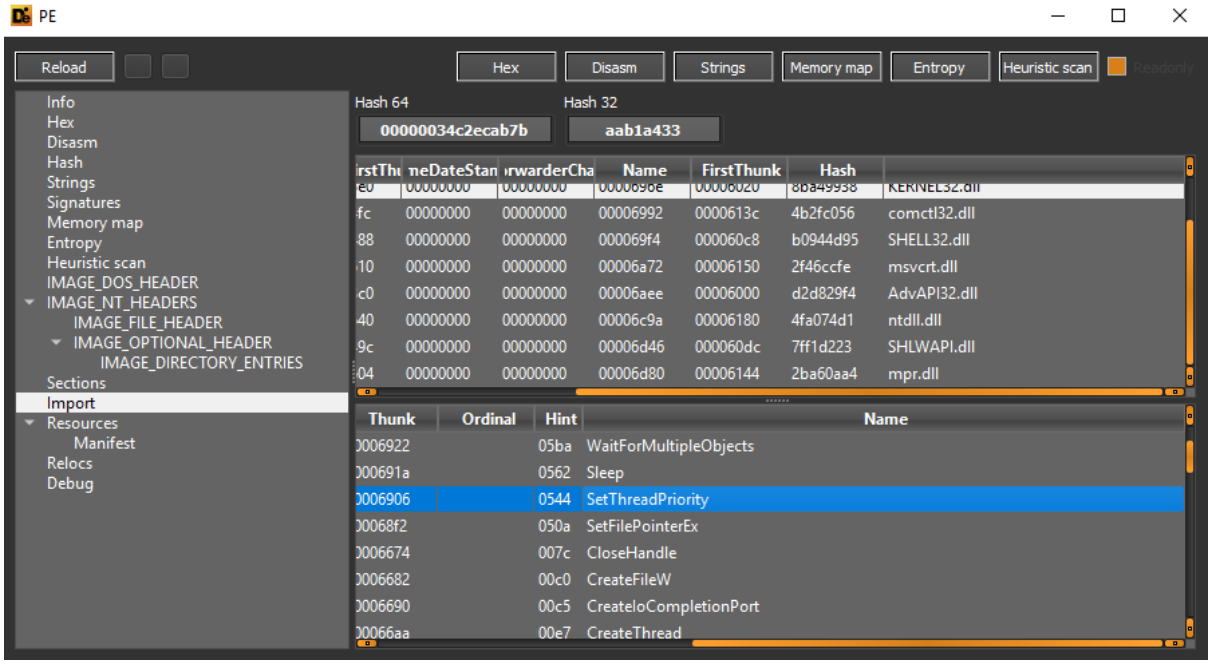
Here are the details of the import of mpr.dll for the functions WNetAddConnection2W and WNetGetUniversalNameW:

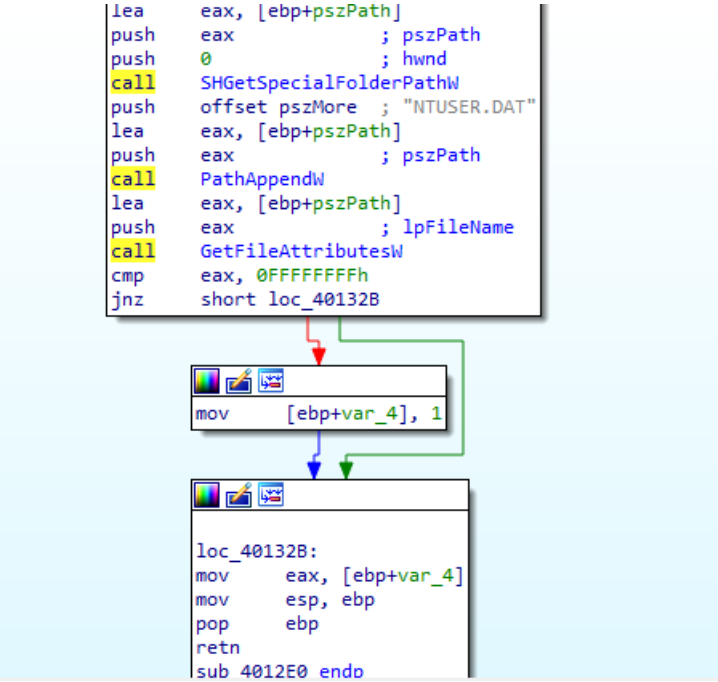



PE
Reload
Hex
Disasm
Strings
Memory map
Entropy
Heuristic scan
Readonly
Info
Hex
Disasm
Hash
Strings
Signatures
Memory map
Entropy
Heuristic scan
IMAGE_DOS_HEADER
IMAGE_NT_HEADERS
IMAGE_FILE_HEADER
IMAGE_OPTIONAL_HEADER
IMAGE_DIRECTORY_ENTRIES
Sections
Import
Resources
Manifest
Relocs
Debug
Hash 64
00000034c2ecab7b
Hash 32
aab1a433
ginalFirstTh
meDateStan
rwarderCha
Name
FirstThunk
Hash
2 000064fc 00000000 00000000 00006992 0000613c 4b2fc056 comctl32.dll
3 00006488 00000000 00000000 000069f4 000060c8 b0944d95 SHELL32.dll
4 00006510 00000000 00000000 00006a72 00006150 2f46ccfe msvcrt.dll
5 000063c0 00000000 00000000 00006aee 00006000 d2d829f4 AdvAPI32.dll
6 00006540 00000000 00000000 00006c9a 00006180 4fa074d1 ntdll.dll
7 0000649c 00000000 00000000 00006d46 000060dc 7ff1d223 SHLWAPI.dll
8 00006504 00000000 00000000 00006d80 00006144 2ba60aa4 mpr.dll
Thunk
Ordinal
Hint
Name
1 00006cee 005a PathIsDirectoryEmptyW
2 00006cc4 0047 PathFindExtensionW
3 00006cb2 0045 PathFileExistsW
4 00006d1a 0061 PathIsNetworkPathW
5 00006d06 005b PathIsDirectoryW
6 00006d30 008b PathRemoveFileSpecW
7 00006ca4 0034 PathAppendW

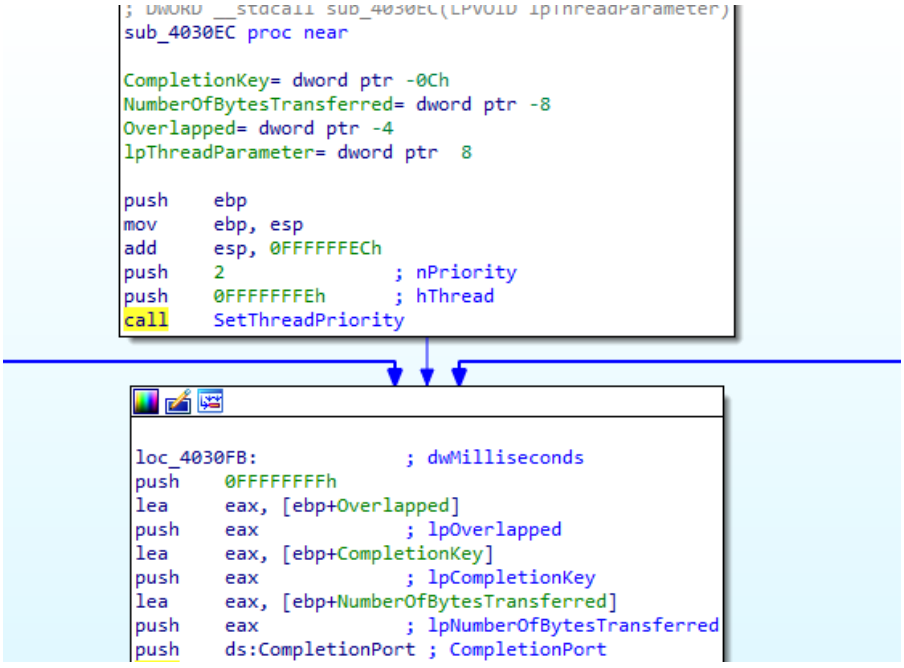

PE
Reload
Hex
Disasm
Strings
Memory map
Entropy
Heuristic scan
Readonly
Info
Hex
Disasm
Hash
Strings
Signatures
Memory map
Entropy
Heuristic scan
IMAGE_DOS_HEADER
IMAGE_NT_HEADERS
IMAGE_FILE_HEADER
IMAGE_OPTIONAL_HEADER
IMAGE_DIRECTORY_ENTRIES
Sections
Import
Resources
Manifest
Relocs
Debug
Hash 64
00000034c2ecab7b
Hash 32
aab1a433
Name
FirstThunk
Hash
2 000064fc 00000000 00000000 00006992 0000613c 4b2fc056 comctl32.dll
3 00006488 00000000 00000000 000069f4 000060c8 b0944d95 SHELL32.dll
4 00006510 00000000 00000000 00006a72 00006150 2f46ccfe msvcrt.dll
5 000063c0 00000000 00000000 00006aee 00006000 d2d829f4 AdvAPI32.dll
6 00006540 00000000 00000000 00006c9a 00006180 4fa074d1 ntdll.dll
7 0000649c 00000000 00000000 00006d46 000060dc 7ff1d223 SHLWAPI.dll
8 00006504 00000000 00000000 00006d80 00006144 2ba60aa4 mpr.dll
Thunk
Ordinal
Hint
Name
0 00006aae 0000 MD5Update
1 00006aa4 0000 MD5Init
2 00006a98 0000 MD5Final
3 00006a7e 0000 ConvertSidToStringSidW
4 00006acc 0000 RegDeleteKeyW
5 00006adc 0000 RegSetValueExW
6 00006aba 0000 RegCreateKeyExW

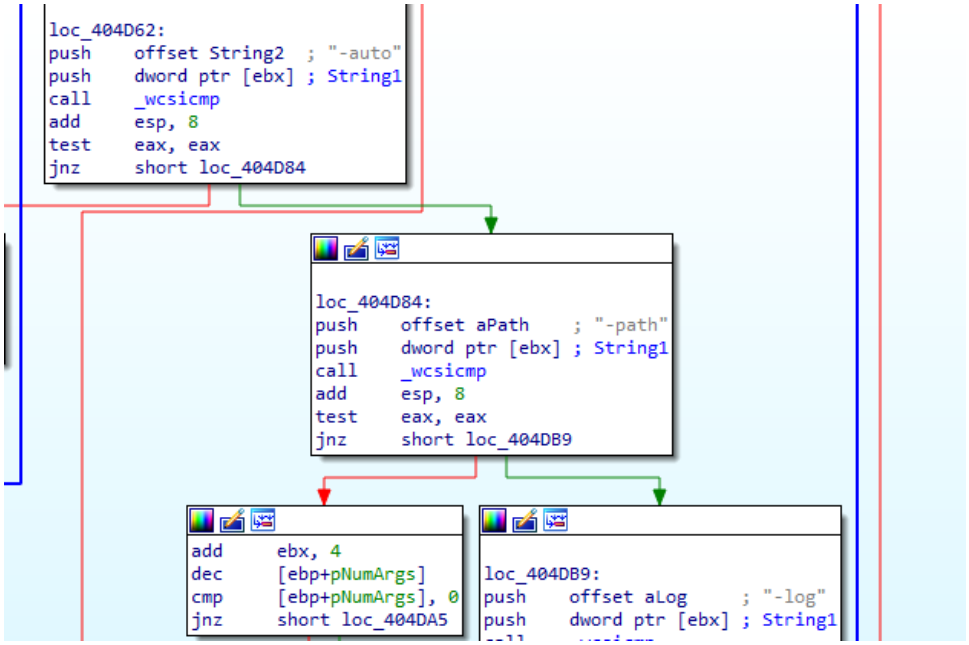

PE
Reload
Hex
Disasm
Strings
Memory map
Entropy
Heuristic scan
Readonly
Info
Hex
Disasm
Hash
Strings
Signatures
Memory map
Entropy
Heuristic scan
IMAGE_DOS_HEADER
IMAGE_NT_HEADERS
IMAGE_FILE_HEADER
IMAGE_OPTIONAL_HEADER
IMAGE_DIRECTORY_ENTRIES
Sections
Import
Resources
Manifest
Relocs
Debug
Hash 64
00000034c2ecab7b
Hash 32
aab1a433
ginalFirstThu neDateStan rwarderCha Name FirstThunk Hash
0 000064c0 00000000 00000000 00006668 00006100 6653d947 USER32.dll
1 000063e0 00000000 00000000 0000696e 00006020 8ba49938 KERNEL32.dll
2 000064fc 00000000 00000000 00006992 0000613c 4b2fc056 comctl32.dll
3 00006488 00000000 00000000 000069f4 000060c8 b0944d95 SHELL32.dll
4 00006510 00000000 00000000 00006a72 00006150 2f46ccfe msvcrt.dll
5 000063c0 00000000 00000000 00006aee 00006000 d2d829f4 AdvAPI32.dll
6 00006540 00000000 00000000 00006c9a 00006180 4fa074d1 ntdll.dll
Thunk Ordinal Hint Name
0 000069da 0175 SHGetSpecialFolderPathW
1 000069a0 0007 CommandLineToArgvW
2 000069c8 008b SHChangeNotify
3 000069b6 0028 DragQueryFileW


PE
Reload
Hex
Disasm
Strings
Memory map
Entropy
Heuristic scan
Readonly
Info
Hex
Disasm
Hash
Strings
Signatures
Memory map
Entropy
Heuristic scan
IMAGE_DOS_HEADER
IMAGE_NT_HEADERS
IMAGE_FILE_HEADER
IMAGE_OPTIONAL_HEADER
IMAGE_DIRECTORY_ENTRIES
Sections
Import
Resources
Manifest
Relocs
Debug
Hash 64
00000034c2ecab7b
Hash 32
aab1a433
Thunk Ordinal Hint Name
0006922 05ba WaitForMultipleObjects
000691a 0562 Sleep
0006906 0544 SetThreadPriority
00068f2 050a SetFilePointerEx
0006674 007c CloseHandle
0006682 00c0 CreateFileW
0006690 00c5 CreateIoCompletionPort
00066aa 00e7 CreateThread

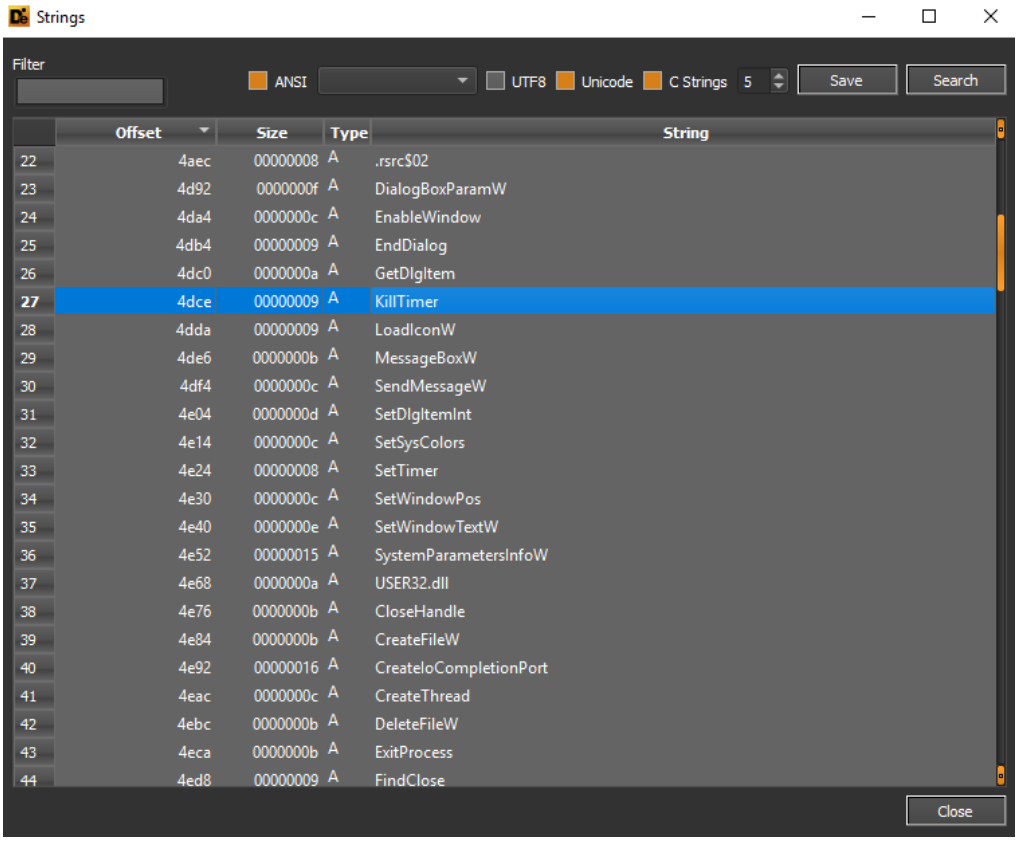


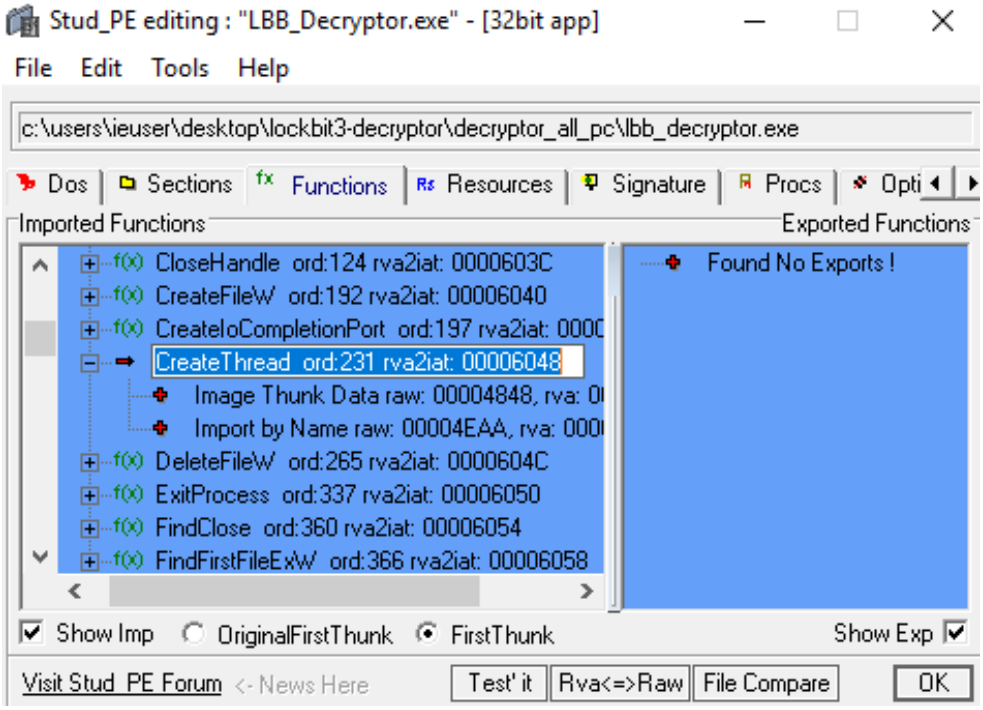
Contextually to the access to the file **NTUSER.dat** with the function **sub_4012E0** is possible to see how execution parameters have been set, included **"-log"** option:

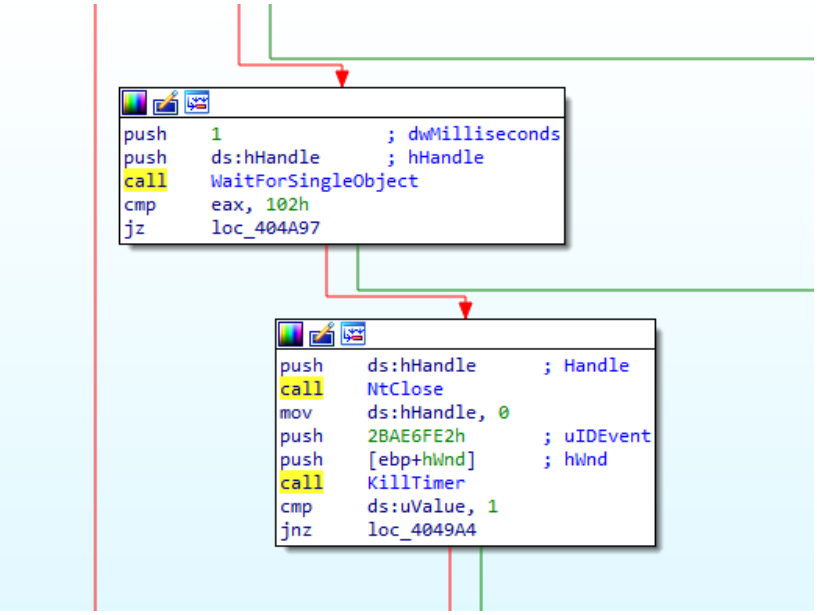




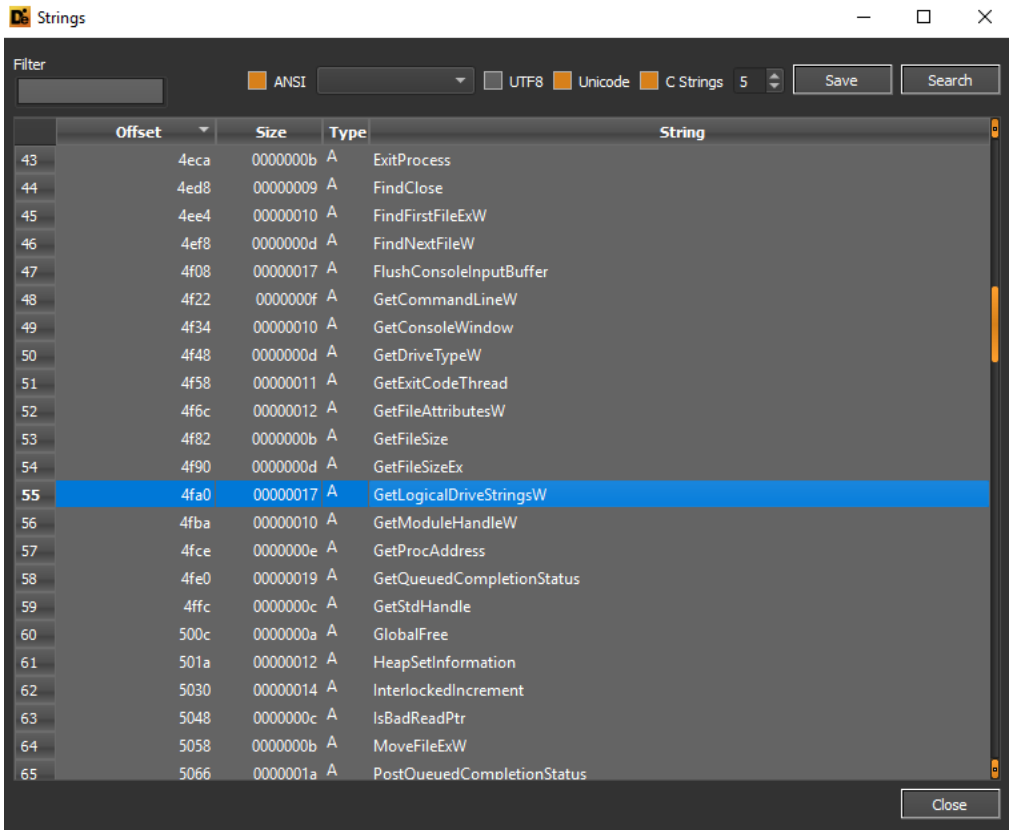
By analyzing the extractable strings from the executable is possible to highlight the peculiarity to stop a timer (through the KillTimer function), the enumeration of the encrypted files through cycles and the use of the functions FindFirstFileExW, FindNextFileW, GetLogicalDrivesW and GetDriveTypeW which are called to manage the drives of the machine on which the decryptor is executed. Furthermore, some modifications of the files attribute are performed through the execution of the function SetFileAttributesW and, for efficiency, to point to the files that are managed, it is executed the function SetFilePointerEx.


Strings
Filter
ANSI
UTF8
Unicode
C Strings
5
Save
Search
Offset
Size
Type
String
22 4aec 00000008 A .rsrc$02
23 4d92 0000000f A DialogBoxParamW
24 4da4 0000000c A EnableWindow
25 4db4 00000009 A EndDialog
26 4dc0 0000000a A GetDlgItem
27 4dce 00000009 A KillTimer
28 4dda 00000009 A LoadIconW
29 4de6 0000000b A MessageBoxW
30 4df4 0000000c A SendMessageW
31 4e04 0000000d A SetDlgItemInt
32 4e14 0000000c A SetSysColors
33 4e24 00000008 A SetTimer
34 4e30 0000000c A SetWindowPos
35 4e40 0000000e A SetWindowTextW
36 4e52 00000015 A SystemParametersInfoW
37 4e68 0000000a A USER32.dll
38 4e76 0000000b A CloseHandle
39 4e84 0000000b A CreateFileW
40 4e92 00000016 A CreateIoCompletionPort
41 4eac 0000000c A CreateThread
42 4ebc 0000000b A DeleteFileW
43 4eca 0000000b A ExitProcess
44 4ed8 00000009 A FindClose
Close


Stud_PE editing : "LBB_Decryptor.exe" - [32bit app]
File Edit Tools Help
c:\users\ieuser\desktop\lockbit3-decryptor\decryptor_all_pc\lbb_decryptor.exe
Dos Sections Functions Resources Signature Procs Opti
Imported Functions
Exported Functions
CloseHandle ord:124 rva2iat: 0000603C
CreateFileW ord:192 rva2iat: 00006040
CreateIoCompletionPort ord:197 rva2iat: 0000
CreateThread ord:231 rva2iat: 00006048
Image Thunk Data raw: 00004848, rva: 0
Import by Name raw: 00004EAA, rva: 000
DeleteFileW ord:265 rva2iat: 0000604C
ExitProcess ord:337 rva2iat: 00006050
FindClose ord:360 rva2iat: 00006054
FindFirstFileExW ord:366 rva2iat: 00006058
Found No Exports !
Show Imp
OriginalFirstThunk
FirstThunk
Show Exp
Visit Stud_PE Forum <- News Here
Test' it
Rva<=>Raw
File Compare
OK



Here the extraction of the string related to the function GetLogicalDriveStringsW, which fills a strings buffer with the detected drives on the host:



Strings

Filter

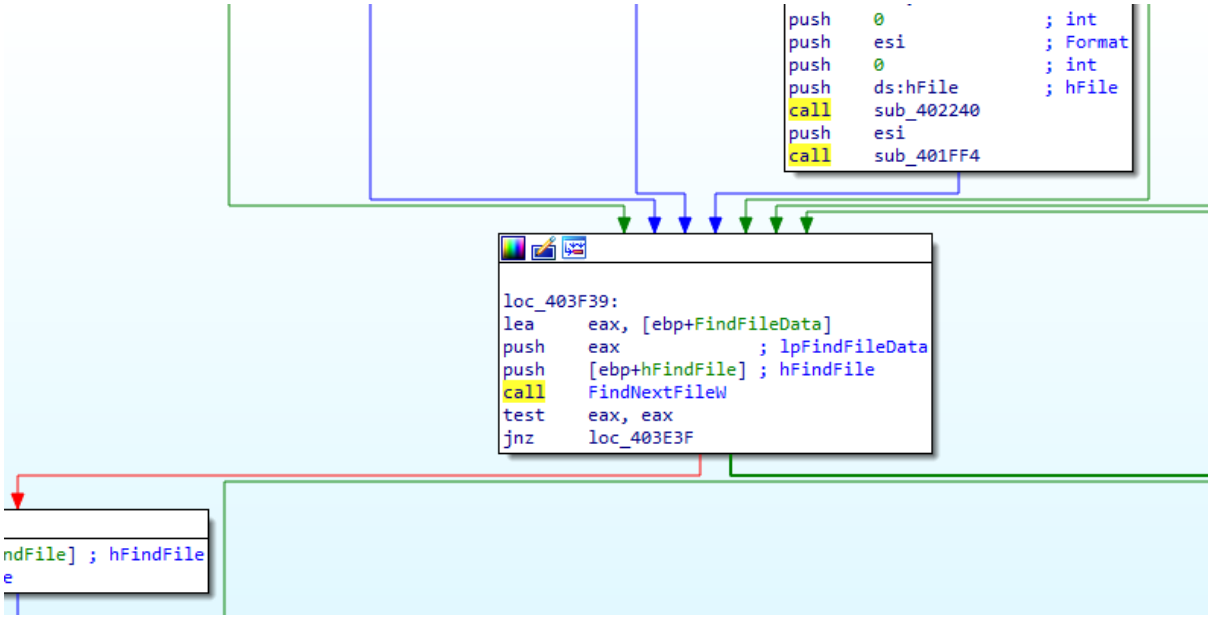
ANSI UTF8 Unicode C Strings 5 Save Search

	Offset	Size	Type	String
64	5058	0000000b	A	MoveFileExW
65	5066	0000001a	A	PostQueuedCompletionStatus
66	5084	00000008	A	ReadFile
67	5090	0000000c	A	ResumeThread
68	50a0	00000017	A	SetConsoleTextAttribute
69	50ba	00000010	A	SetConsoleTitleW
70	50ce	0000000c	A	SetEndOfFile
71	50de	00000012	A	SetFileAttributesW
72	50f4	00000010	A	SetFilePointerEx
73	5108	00000011	A	SetThreadPriority
74	511c	00000005	A	Sleep
75	5124	00000016	A	WaitForMultipleObjects
76	513e	00000013	A	WaitForSingleObject
77	5154	0000000d	A	WriteConsoleW
78	5164	00000009	A	WriteFile
79	516e	0000000c	A	KERNEL32.dll
80	517e	00000012	A	InitCommonControls
81	5192	0000000c	A	comctl32.dll
82	51a2	00000012	A	CommandLineToArgvW
83	51b8	0000000e	A	DragQueryFileW
84	51ca	0000000e	A	SHChangeNotify
85	51dc	00000017	A	SHGetSpecialFolderPathW
86	51f4	0000000b	A	SHELL32.dll

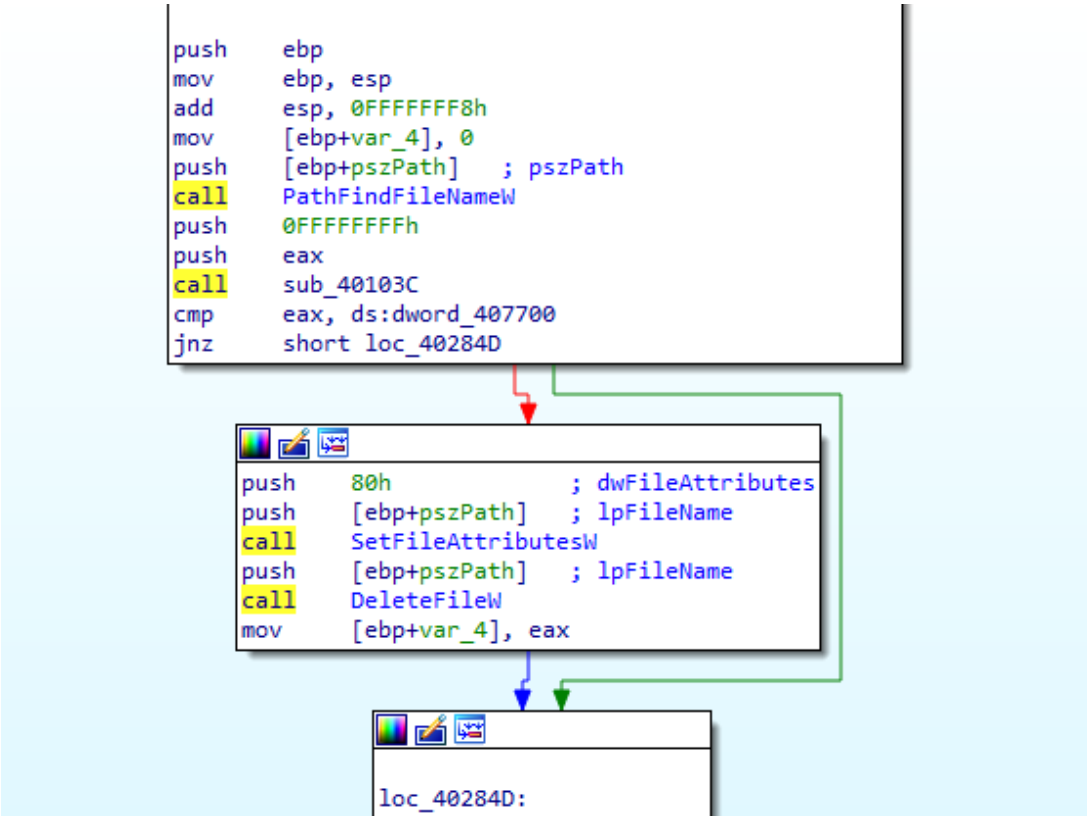
Close

```
loc_403DF5:             ; String
push    [ebp+String]
call    sub_403BE4
mov     [ebp+var_14], eax
push    [ebp+String]
call    sub_401FF4
push    [ebp+dwAdditionalFlags] ; dwAdditionalFlags
push    0               ; lpSearchFilter
push    0               ; fSearchOp
lea     eax, [ebp+FindFileData]
push    eax             ; lpFindFileData
push    0               ; fInfoLevelId
push    [ebp+var_14]    ; lpFileName
call    FindFirstFileExW
mov     [ebp+hFindFile], eax
cmp     [ebp+hFindFile], 0FFFFFFFFh
jz      loc_403F58
```

```
push    2Ah ; '*'       ; Ch
push    [ebp+var_14]    ; Str
call    wcsrchr
add     esp, 8
```




```
; Attributes: bp-based frame

; DWORD __stdcall sub_402CE4(LPVOID lpThreadParameter)
sub_402CE4 proc near

lpThreadParameter= dword ptr  8

push    ebp
mov     ebp, esp
mov     eax, [ebp+lpThreadParameter]
push    dword ptr [eax+4] ; lpBuffer
push    dword ptr [eax] ; nBufferLength
call    GetLogicalDriveStringsW
pop     ebp
retn    4
sub_402CE4 endp
```

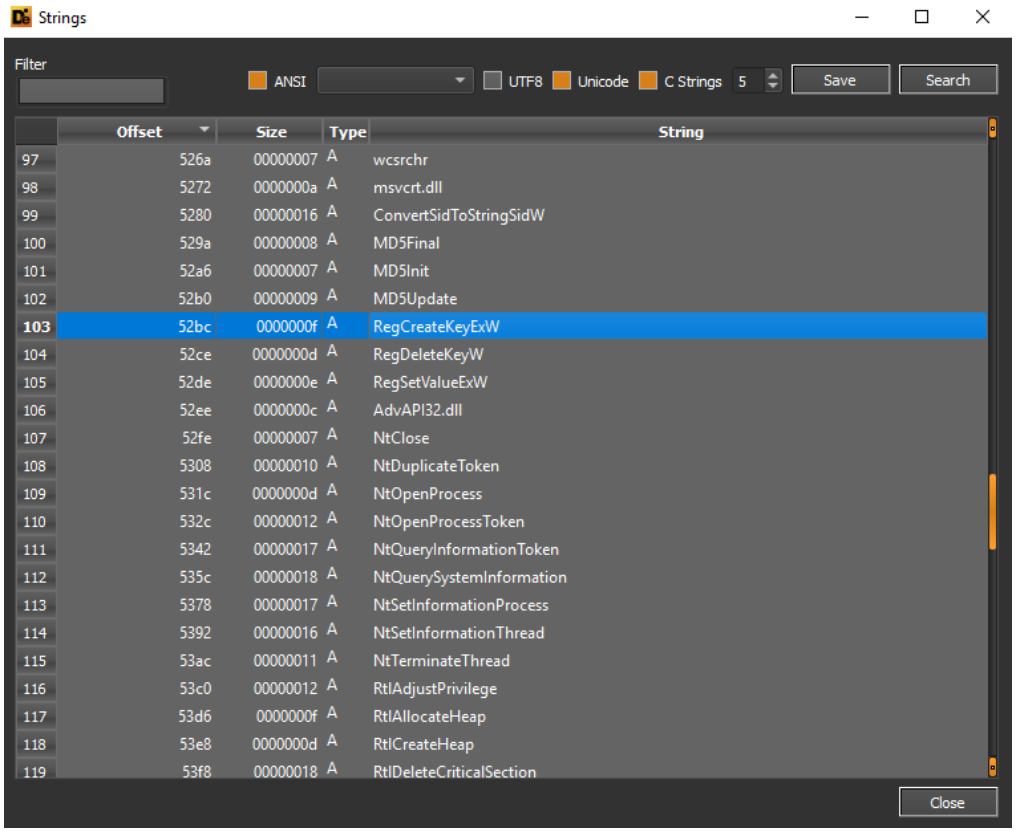


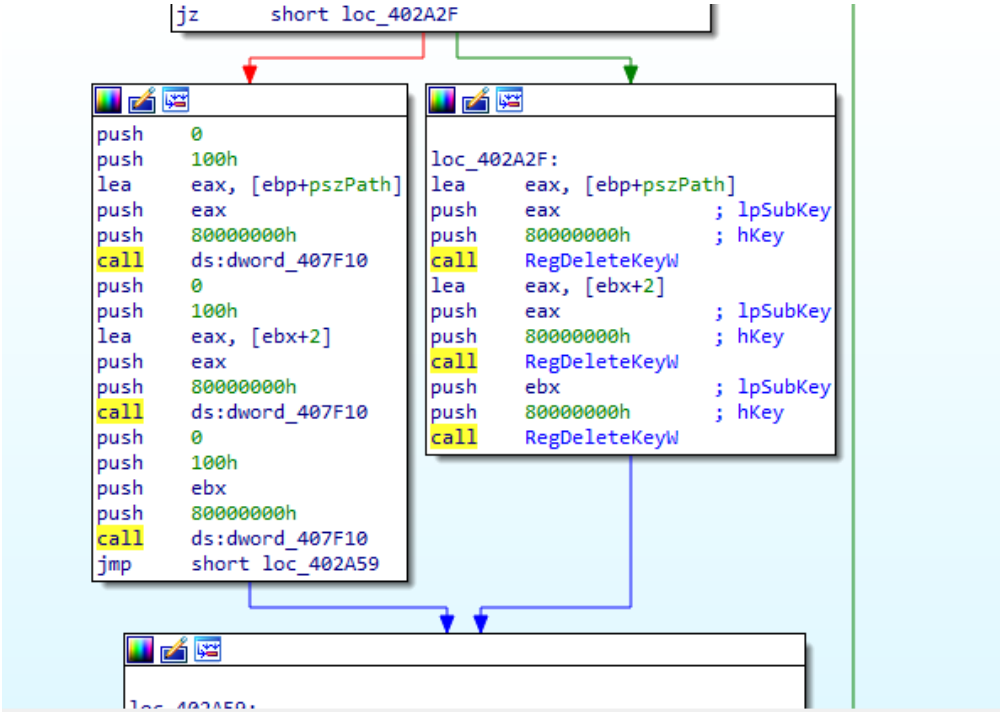
Following, the evidence of use of the library **AdvAPI32.dll** for the reasons mentioned before. It is fundamental to highlight the presence of the three main functions of the MD5 digest phase for the README file:

MD5Init -> Initialize the MD5 digesting context

MD5Update -> Partially generate the digest by taking in consideration some bytes pointed from the input and it updates the hashing digest context

MD5Final -> Concludes the MD5 digesting generation

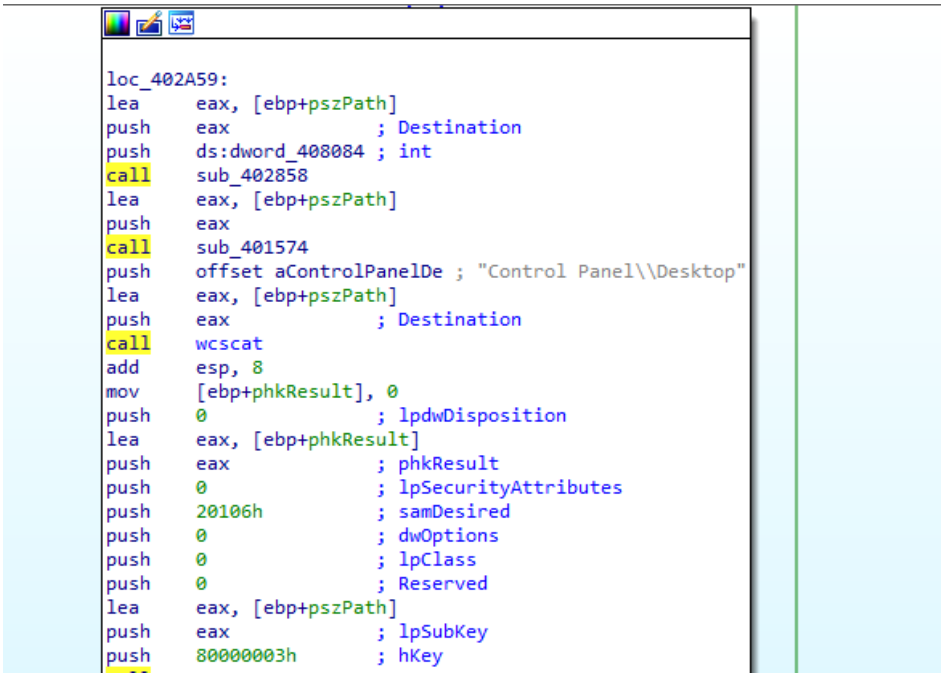

Strings
Filter
ANSI
UTF8
Unicode
C Strings
5
Save
Search
Offset
Size
Type
String
97 526a 00000007 A wcsrchr
98 5272 0000000a A msvcrt.dll
99 5280 00000016 A ConvertSidToStringSidW
100 529a 00000008 A MD5Final
101 52a6 00000007 A MD5Init
102 52b0 00000009 A MD5Update
103 52bc 0000000f A RegCreateKeyExW
104 52ce 0000000d A RegDeleteKeyW
105 52de 0000000e A RegSetValueExW
106 52ee 0000000c A AdvAPI32.dll
107 52fe 00000007 A NtClose
108 5308 00000010 A NtDuplicateToken
109 531c 0000000d A NtOpenProcess
110 532c 00000012 A NtOpenProcessToken
111 5342 00000017 A NtQueryInformationToken
112 535c 00000018 A NtQuerySystemInformation
113 5378 00000017 A NtSetInformationProcess
114 5392 00000016 A NtSetInformationThread
115 53ac 00000011 A NtTerminateThread
116 53c0 00000012 A RtlAdjustPrivilege
117 53d6 0000000f A RtlAllocateHeap
118 53e8 0000000d A RtlCreateHeap
119 53f8 00000018 A RtlDeleteCriticalSection
Close


jz short loc_402A2F
push 0
push 100h
lea eax, [ebp+pszPath]
push eax
push 80000000h
call ds:dword_407F10
push 0
push 100h
lea eax, [ebx+2]
push eax
push 80000000h
call ds:dword_407F10
push 0
push 100h
push ebx
push 80000000h
call ds:dword_407F10
jmp short loc_402A59
loc_402A2F:
lea eax, [ebp+pszPath]
push eax ; lpSubKey
push 80000000h ; hKey
call RegDeleteKeyW
lea eax, [ebx+2]
push eax ; lpSubKey
push 80000000h ; hKey
call RegDeleteKeyW
push ebx ; lpSubKey
push 80000000h ; hKey
call RegDeleteKeyW

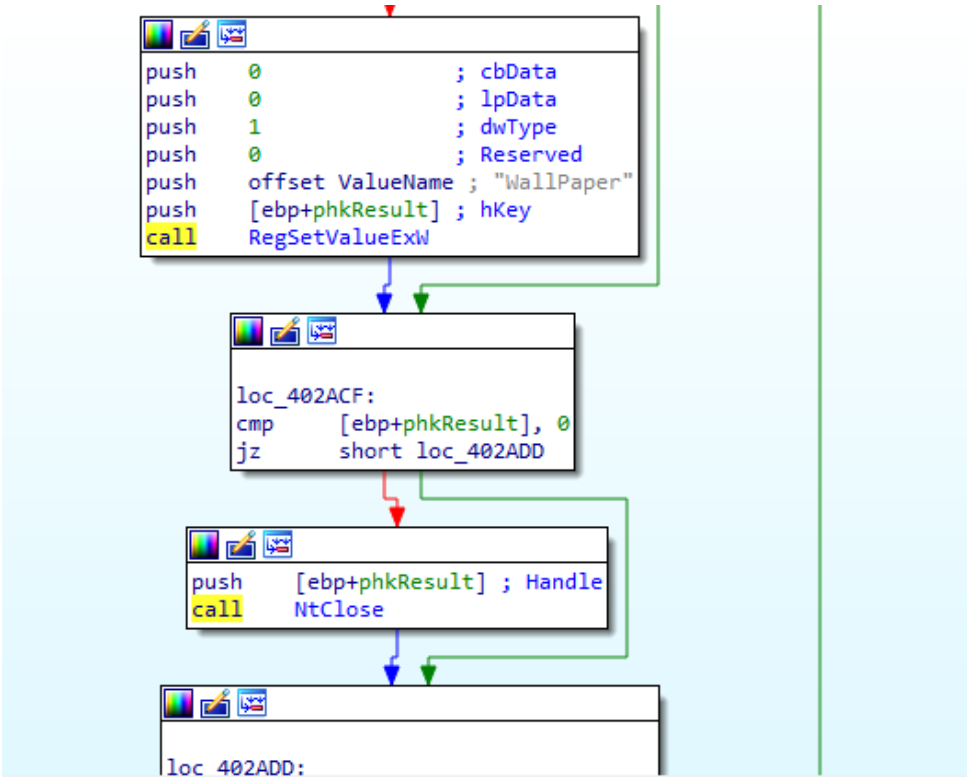
```
; int __stdcall sub_402600(wchar_t *Buffer)
sub_402600 proc near

var_68= byte ptr -68h
var_10= byte ptr -10h
Buffer= dword ptr  8

push    ebp
mov     ebp, esp
add     esp, 0FFFFFF98h
push    ebx
lea     eax, [ebp+var_68]
push    eax
call    MD5Init
mov     ecx, ds:dword_407704
push    80h
lea     eax, [ecx+80h]
push    eax
lea     eax, [ebp+var_68]
push    eax
call    MD5Update
lea     eax, [ebp+var_68]
push    eax
call    MD5Final
lea     ebx, [ebp+var_10]
movzx   eax, byte ptr [ebx+0Fh]
```

Desktop wallpaper and the icons are restored by manipulating the registry keys associated to them:

```
call    DeleteFileW
lea     eax, [ebp+pszPath]
push    eax             ; pszPath
call    PathFindExtensionW
push    offset aIco     ; ".ico"
push    eax             ; Destination
call    wcscpy
add     esp, 8
lea     eax, [ebp+pszPath]
push    eax             ; lpFileName
call    DeleteFileW
lea     eax, [ebx+2]
push    eax             ; Source
lea     eax, [ebp+pszPath]
push    eax             ; Destination
call    wcscpy
add     esp, 8
push    offset aDefaulticon ; "\\DefaultIcon"
lea     eax, [ebp+pszPath]
push    eax             ; Destination
call    wcscat
add     esp, 8
push    offset ModuleName ; "advapi32.dll"
call    GetModuleHandleW
push    offset ProcName ; "RegDeleteKeyExW"
push    eax             ; hModule
call    GetProcAddress
mov     ds:dword_407F10, eax
```



```
loc_402A59:
lea     eax, [ebp+pszPath]
push    eax             ; Destination
push    ds:dword_408084 ; int
call    sub_402858
lea     eax, [ebp+pszPath]
push    eax
call    sub_401574
push    offset aControlPanelDe ; "Control Panel\\Desktop"
lea     eax, [ebp+pszPath]
push    eax             ; Destination
call    wcscat
add     esp, 8
mov     [ebp+phkResult], 0
push    0               ; lpdwDisposition
lea     eax, [ebp+phkResult]
push    eax             ; phkResult
push    0               ; lpSecurityAttributes
push    20106h          ; samDesired
push    0               ; dwOptions
push    0               ; lpClass
push    0               ; Reserved
lea     eax, [ebp+pszPath]
push    eax             ; lpSubKey
push    80000003h       ; hKey
```

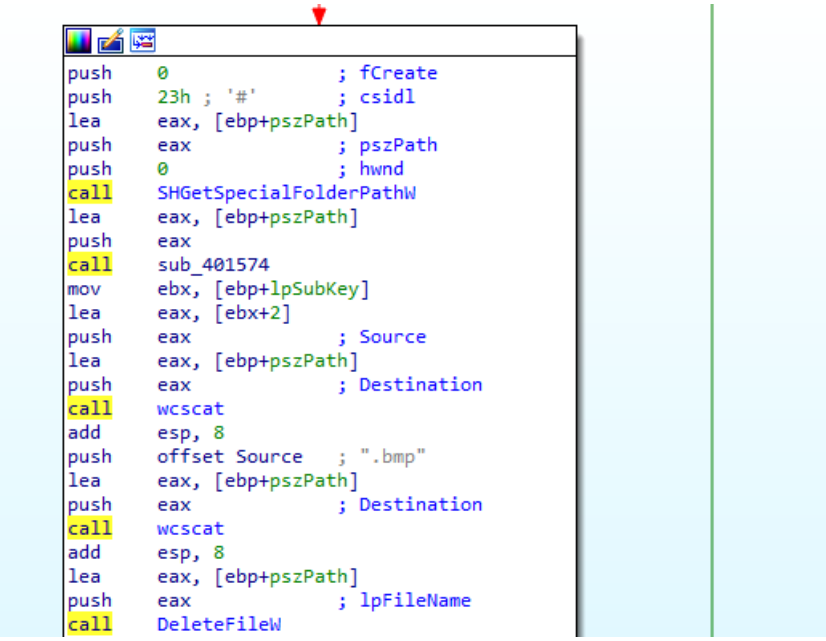


```
push    0                   ; cbData
push    0                   ; lpData
push    1                   ; dwType
push    0                   ; Reserved
push    offset ValueName ; "WallPaper"
push    [ebp+phkResult] ; hKey
call    RegSetValueExW
```

```
loc_402ACF:
cmp     [ebp+phkResult], 0
jz      short loc_402ADD
```

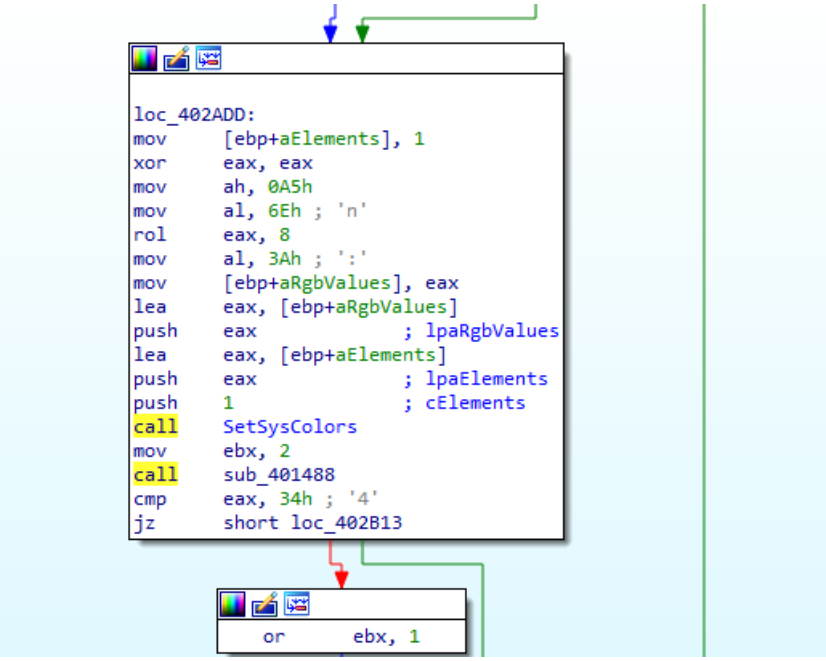
```
push    [ebp+phkResult] ; Handle
call    NtClose
```

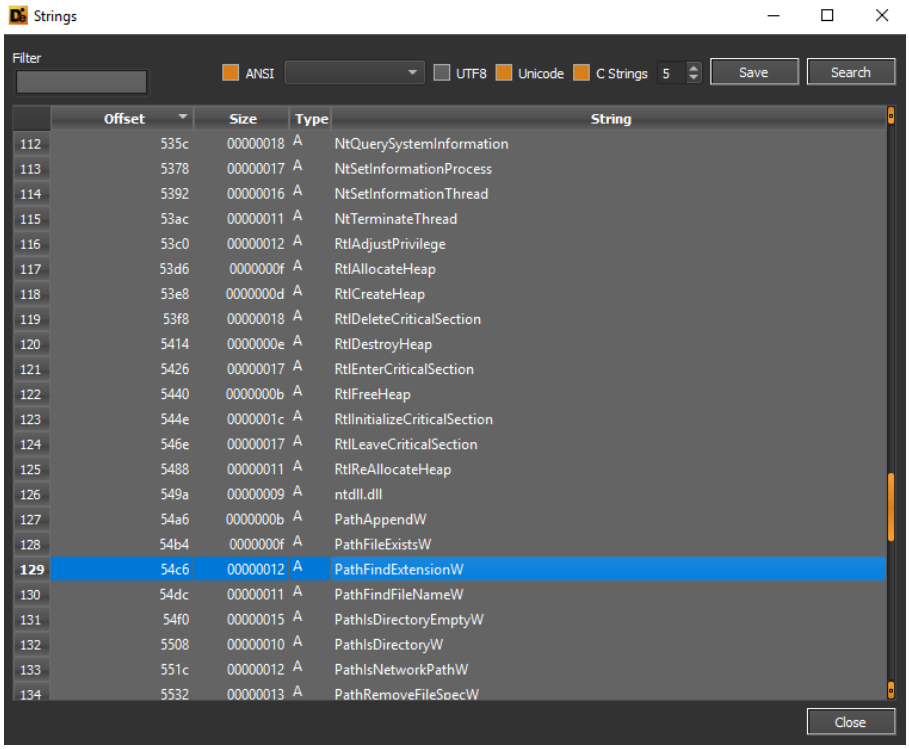
```
loc_402ADD:
```



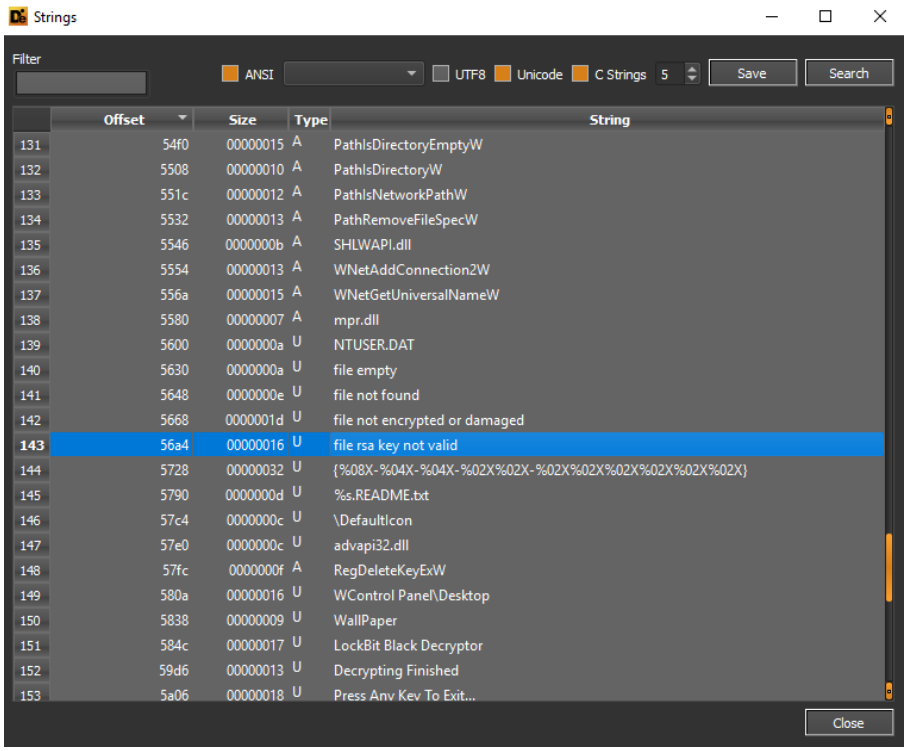
A loop on 4 elements is performed for the RGB settings and the function SetSysColors is called.

This function can be executed to do the colour change of a graphic element:





The analyzed decryptor performs, after creating the thread for efficiency scope, the subroutine of decryption checking if the RSA key is invalid or corrupted. The debugging string ***"file rsa key not valid"*** is indeed present in this case**.**



```
push    eax
movzx   eax, byte ptr [ebx+0Bh]
push    eax
movzx   eax, byte ptr [ebx+0Ah]
push    eax
movzx   eax, byte ptr [ebx+9]
push    eax
movzx   eax, byte ptr [ebx+8]
push    eax
push    small 0
push    small word ptr [ebx+6]
push    small 0
push    small word ptr [ebx+4]
push    dword ptr [ebx] ; Format
push    offset a08x04x04x02x02 ; "{%08X-%04X-%04X-%02X%02X-%02X%02X%02X%0"...
push    [ebp+Buffer]    ; Buffer
call    swprintf
add     esp, 34h
pop     ebx
mov     esp, ebp
pop     ebp
retn    4
sub_402600 endp
```

When the MD5Init function is executed, an empty buffer is used as parameter to be filled with the hash digest when it is actually executed.

```
003F2606    53              push ebx
003F2607    8D45 98         lea eax,dword ptr ss:[ebp-68]
003F260A    50              push eax
003F260B    E8 822B0000     call <JMP.&MD5Init>
003F2610    8B0D 04773F00   mov ecx,dword ptr ds:[3F7704]
003F2616    68 80000000     push 80
003F261B    8D81 80000000   lea eax,dword ptr ds:[ecx+80]
003F2621    50              push eax
003F2622    8D45 98         lea eax,dword ptr ss:[ebp-68]
003F2625    50              push eax
003F2626    E8 6D2B0000     call <JMP.&MD5Update>
003F262B    8D45 98         lea eax,dword ptr ss:[ebp-68]
003F262E    50              push eax
003F262F    E8 582B0000     call <JMP.&MD5Final>
003F2634    8D5D F0         lea ebx,dword ptr ss:[ebp-10]
```

```
Address   Hex                                             ASCII
003F7704  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003F7714  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003F7724  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003F7734  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003F7744  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003F7754  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003F7764  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003F7774  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003F7784  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003F7794  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003F77A4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003F77B4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Subsequently, the general register ECX is used for the execution content and also the EAX register to perform push instructions for the output data of the digest operation:

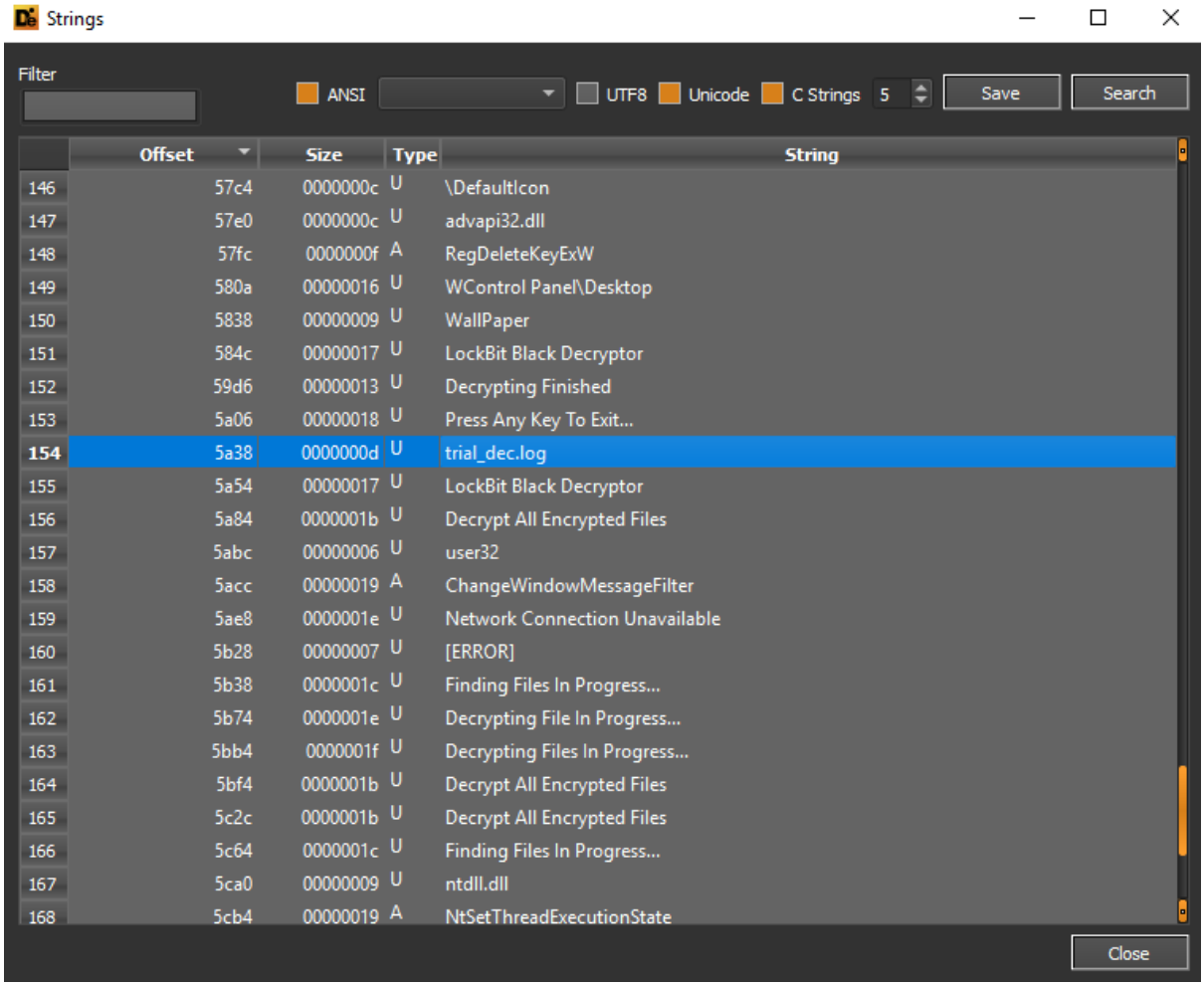
```
003F260B    E8 822B0000     call <JMP.&MD5Init>
003F2610    8B0D 04773F00   mov ecx,dword ptr ds:[3F7704]
003F2616    68 80000000     push 80
003F261B    8D81 80000000   lea eax,dword ptr ds:[ecx+80]
003F2621    50              push eax
003F2622    8D45 98         lea eax,dword ptr ss:[ebp-68]
003F2625    50              push eax
003F2626    E8 6D2B0000     call <JMP.&MD5Update>
003F262B    8D45 98         lea eax,dword ptr ss:[ebp-68]
003F262E    50              push eax
003F262F    E8 582B0000     call <JMP.&MD5Final>
003F2634    8D5D F0         lea ebx,dword ptr ss:[ebp-10]
003F2637    0FB643 0F       movzx eax,byte ptr ds:[ebx+F]
003F263B    50              push eax
003F263C    0FB643 0E       movzx eax,byte ptr ds:[ebx+E]
003F2640    50              push eax
003F2641    0FB643 0D       movzx eax,byte ptr ds:[ebx+D]
003F2645    50              push eax
003F2646    0FB643 0C       movzx eax,byte ptr ds:[ebx+C]
003F264A    50              push eax
003F264B    0FB643 0B       movzx eax,byte ptr ds:[ebx+B]
003F264F    50              push eax
003F2650    0FB643 0A       movzx eax,byte ptr ds:[ebx+A]
003F2654    50              push eax
003F2655    0FB643 09       movzx eax,byte ptr ds:[ebx+9]
003F2659    50              push eax
003F265A    0FB643 08       movzx eax,byte ptr ds:[ebx+8]
003F265E    50              push eax
003F265F    66:6A 00        push 0
003F2662    66:FF73 06      push word ptr ds:[ebx+6]
```

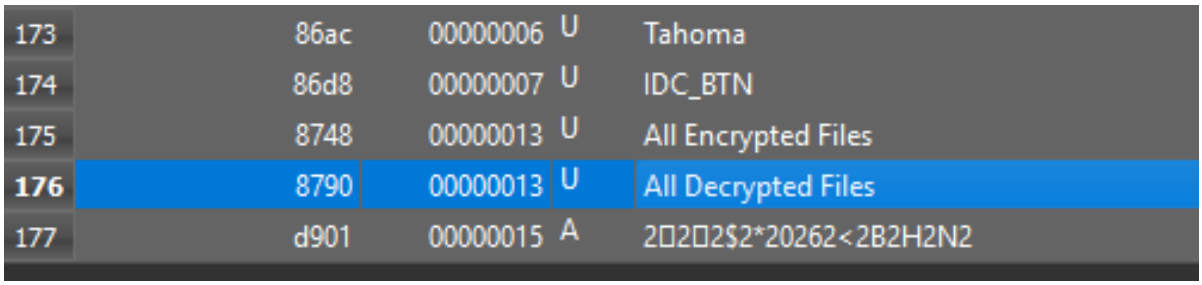
Then, the MD5Final function is used to fill the general ID and the output of the MD5 hash digest:

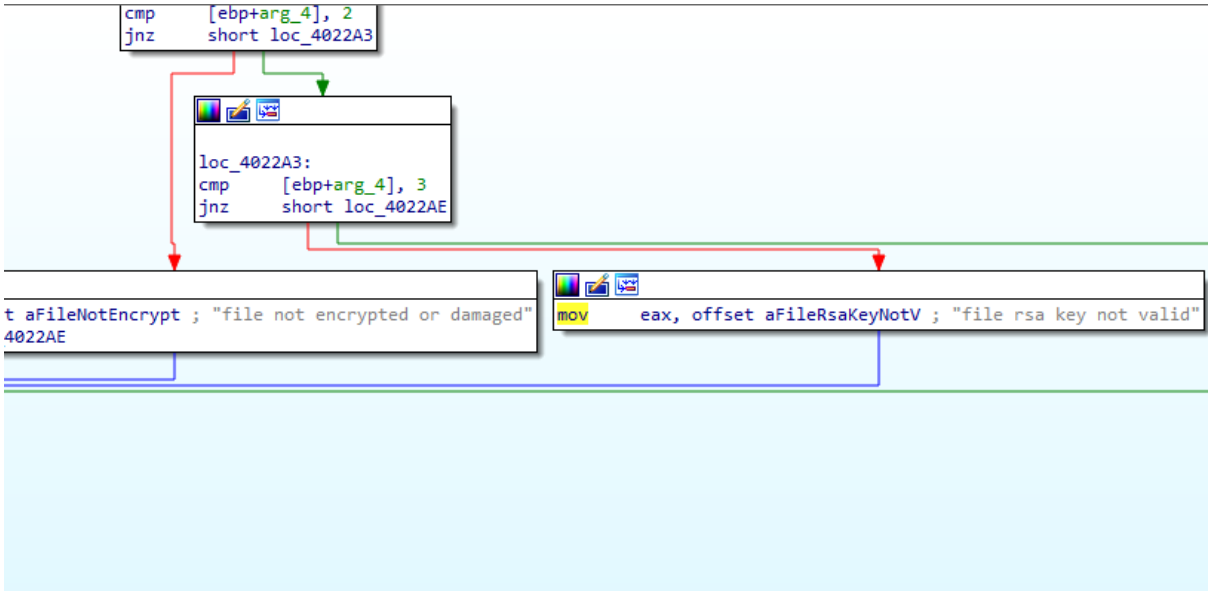
```
push 0
push word ptr ds:[ebx+6]
push 0
push word ptr ds:[ebx+4]
push dword ptr ds:[ebx]
push 1bb_decryptor.3F7128       3F7128:L"{%08X-%04X-%04X-%02X%02X-%02X%02X%02X%02X
push dword ptr ss:[ebp+8]
call <JMP.&swprintf>
add esp,34
pop ebx
mov esp,ebp
pop ebp
ret 4
```

So, it is observable that during the decryption phase, the decryptor tool reads the %s.README.txt file: our tests in lab show that also in absence of the README.txt file, the decryptor manages anyway to decrypt encrypted files. This indicates how in the decryptor tool there are all the information needed to decrypt encrypted files. Each decryptor in fact is generated contextually to its ransomware, starting from a couple of public and private key: the ransomware uses the public key to encrypt, meanwhile the decryptor tool will use the related private key for the decryption.

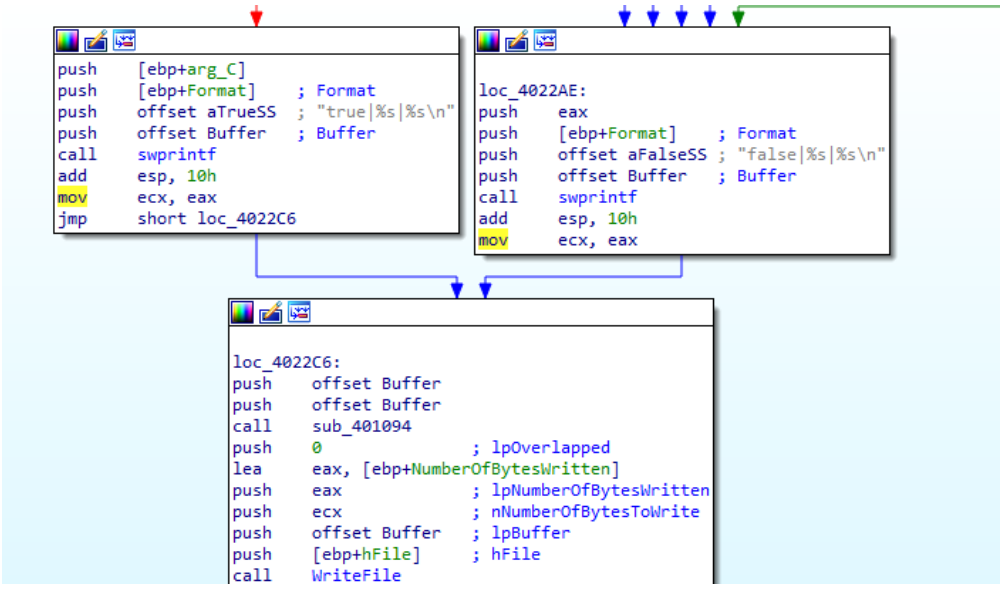
Continuing the decryptor analysis, it is noteable that the execution of the decryption tool is traced also with a log file: **trial_dec.log:**


Strings
Filter
ANSI
UTF8
Unicode
C Strings
5
Save
Search
Offset
Size
Type
String
146 57c4 0000000c U \DefaultIcon
147 57e0 0000000c U advapi32.dll
148 57fc 0000000f A RegDeleteKeyExW
149 580a 00000016 U WControl Panel\Desktop
150 5838 00000009 U WallPaper
151 584c 00000017 U LockBit Black Decryptor
152 59d6 00000013 U Decrypting Finished
153 5a06 00000018 U Press Any Key To Exit...
154 5a38 0000000d U trial_dec.log
155 5a54 00000017 U LockBit Black Decryptor
156 5a84 0000001b U Decrypt All Encrypted Files
157 5abc 00000006 U user32
158 5acc 00000019 A ChangeWindowMessageFilter
159 5ae8 0000001e U Network Connection Unavailable
160 5b28 00000007 U [ERROR]
161 5b38 0000001c U Finding Files In Progress...
162 5b74 0000001e U Decrypting File In Progress...
163 5bb4 0000001f U Decrypting Files In Progress...
164 5bf4 0000001b U Decrypt All Encrypted Files
165 5c2c 0000001b U Decrypt All Encrypted Files
166 5c64 0000001c U Finding Files In Progress...
167 5ca0 00000009 U ntdll.dll
168 5cb4 00000019 A NtSetThreadExecutionState
Close

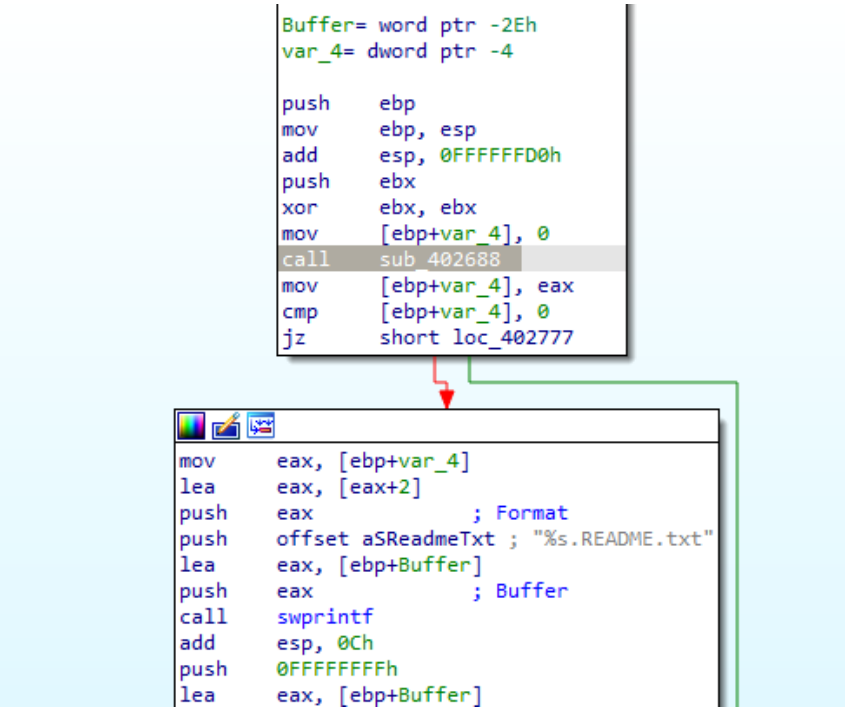

173 86ac 00000006 U Tahoma
174 86d8 00000007 U IDC_BTN
175 8748 00000013 U All Encrypted Files
176 8790 00000013 U All Decrypted Files
177 d901 00000015 A 2□2□2$2*20262<2B2H2N2



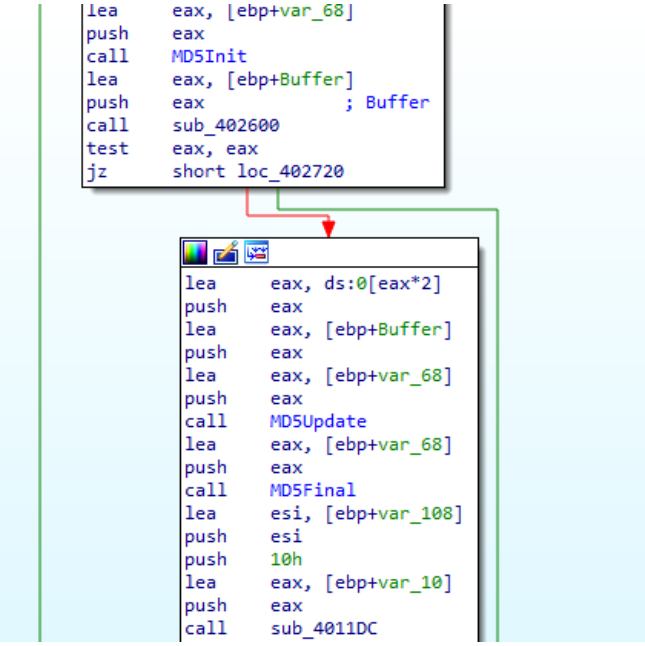
It seems that during the execution the verdicts are written on files for the various operations, with ***"True"*** or ***"False"*** depending on the specific cases.



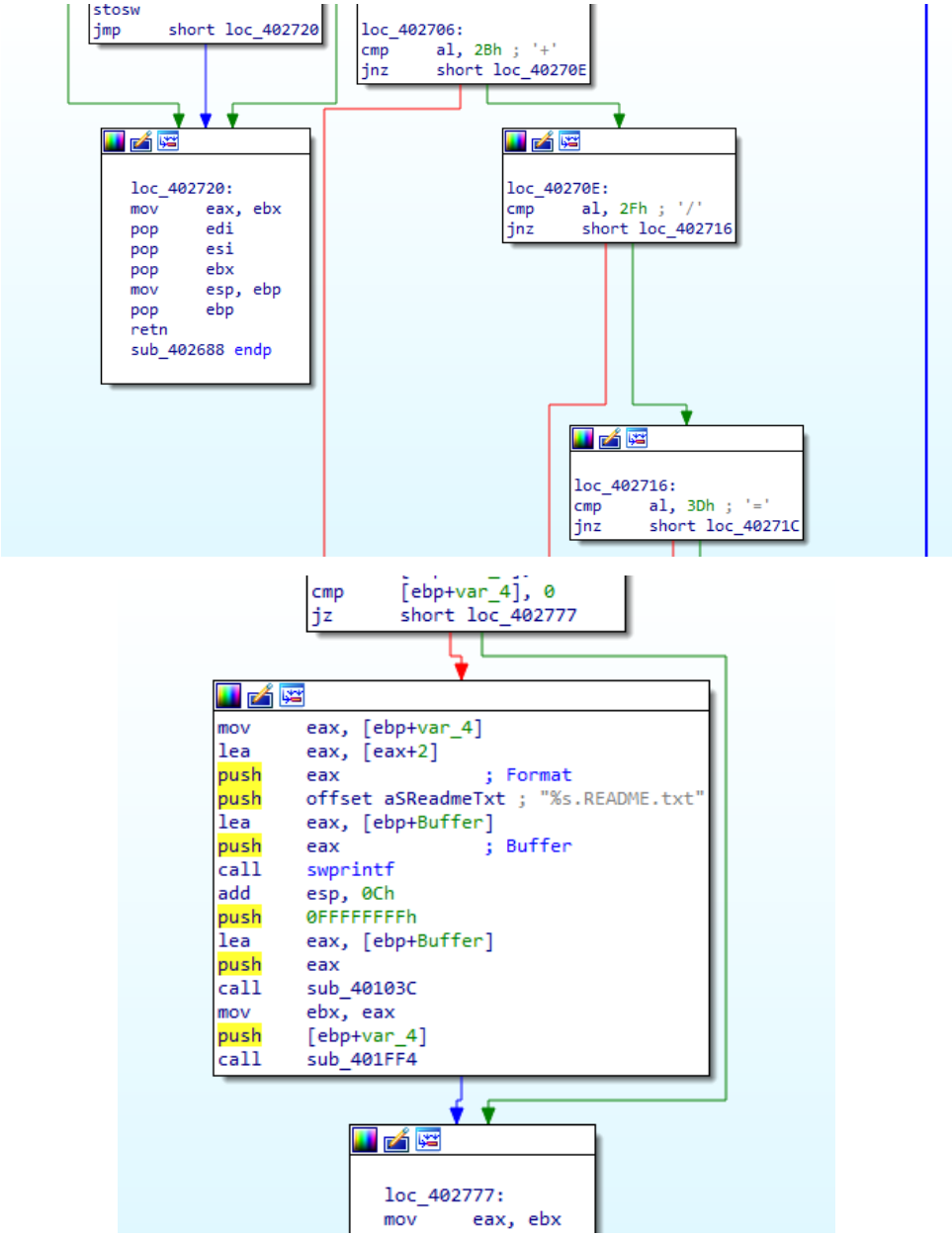
Following are the details of the execution flow which is related to the function sub_402688 for the phase of README.TXT ID hashing.



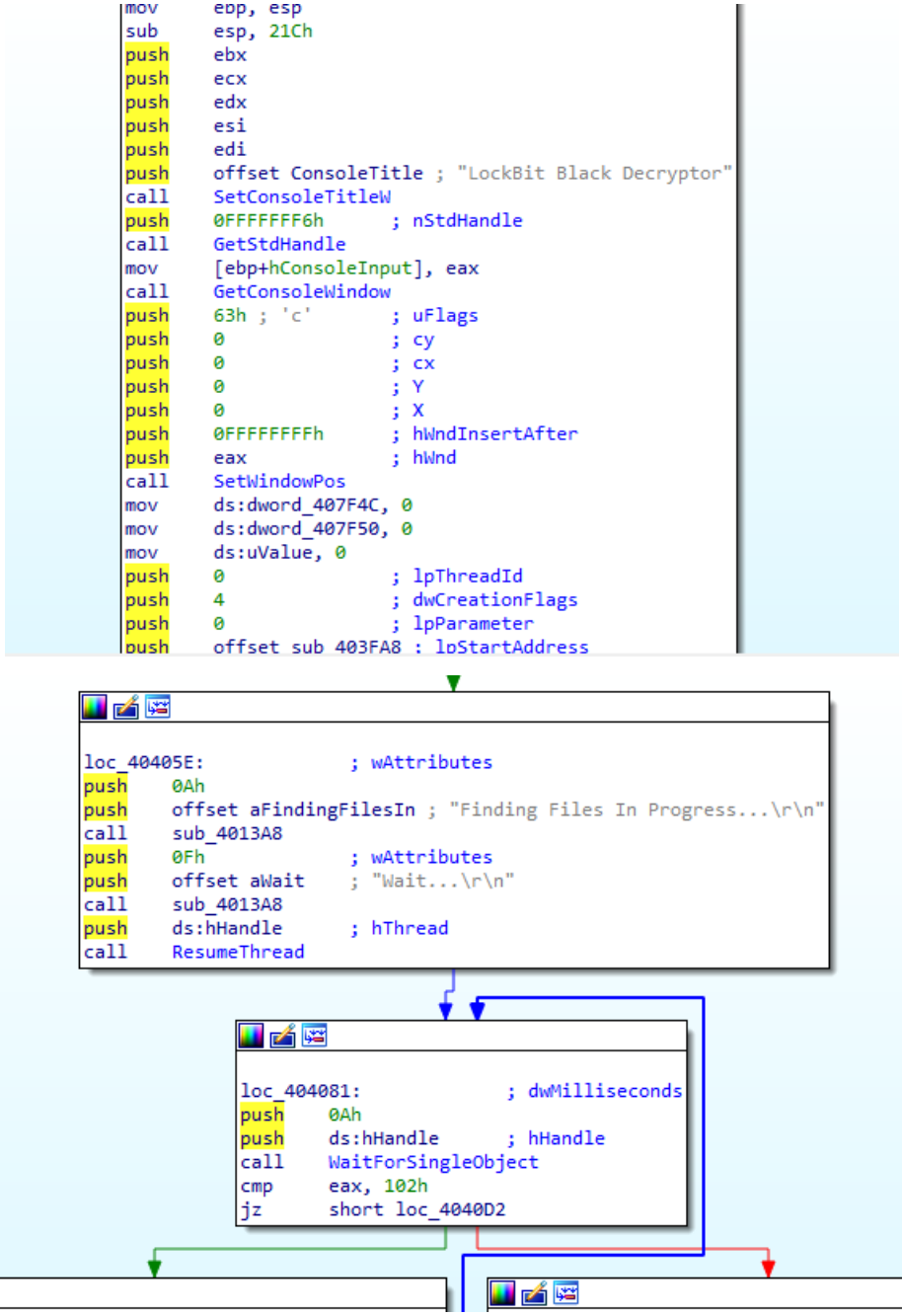
Here are the instances of the creation of the contexts to proceed with the MD5 hashing digest in the context of disassemblying:

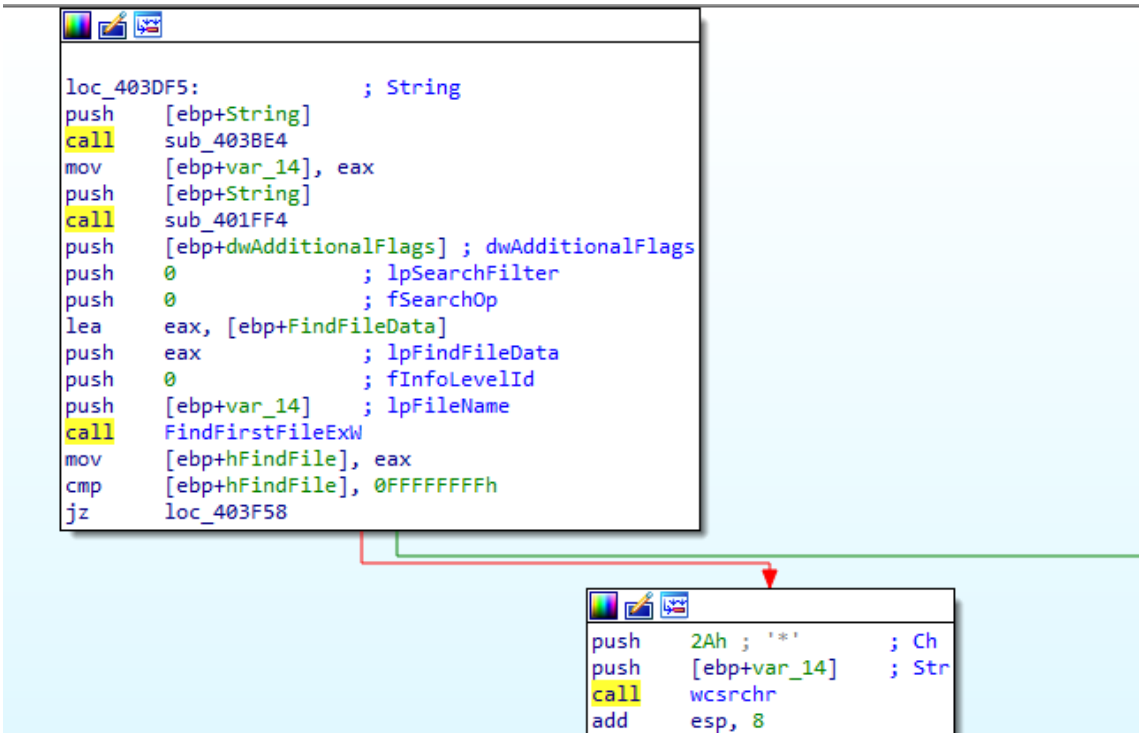


Subsequently it seems that some compare, checking and validation execution are performed:

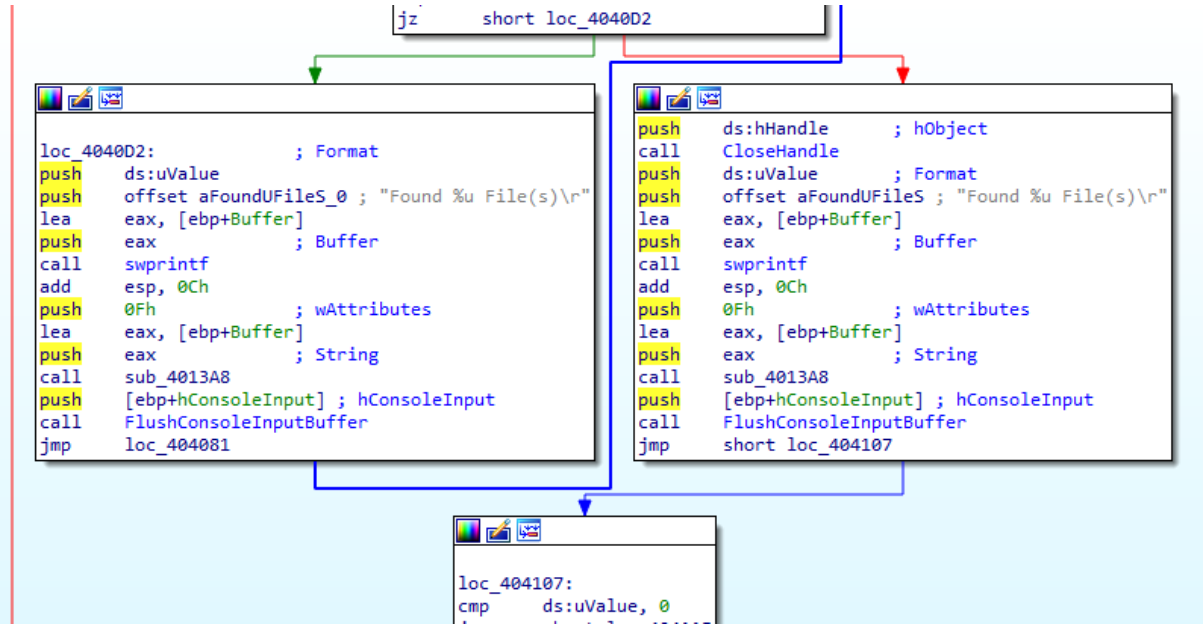


Following, instead, the initial juncture of the execution in which the phase of files search and decryption of them begins (please note that in the files enumeration loop the wildcard * is used):

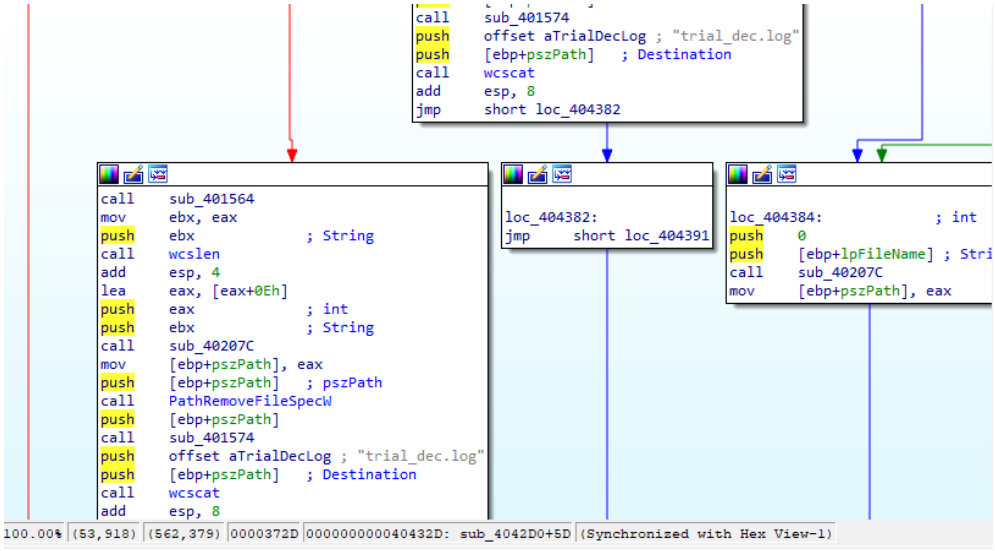

mov ebp, esp
sub esp, 21Ch
push ebx
push ecx
push edx
push esi
push edi
push offset ConsoleTitle ; "LockBit Black Decryptor"
call SetConsoleTitleW
push 0FFFFFFF6h ; nStdHandle
call GetStdHandle
mov [ebp+hConsoleInput], eax
call GetConsoleWindow
push 63h ; 'c' ; uFlags
push 0 ; cy
push 0 ; cx
push 0 ; Y
push 0 ; X
push 0FFFFFFFFh ; hWndInsertAfter
push eax ; hWnd
call SetWindowPos
mov ds:dword_407F4C, 0
mov ds:dword_407F50, 0
mov ds:uValue, 0
push 0 ; lpThreadId
push 4 ; dwCreationFlags
push 0 ; lpParameter
push offset sub_403FA8 ; lpStartAddress
loc_40405E: ; wAttributes
push 0Ah
push offset aFindingFilesIn ; "Finding Files In Progress...\r\n"
call sub_4013A8
push 0Fh ; wAttributes
push offset aWait ; "Wait...\r\n"
call sub_4013A8
push ds:hHandle ; hThread
call ResumeThread
loc_404081: ; dwMilliseconds
push 0Ah
push ds:hHandle ; hHandle
call WaitForSingleObject
cmp eax, 102h
jz short loc_4040D2



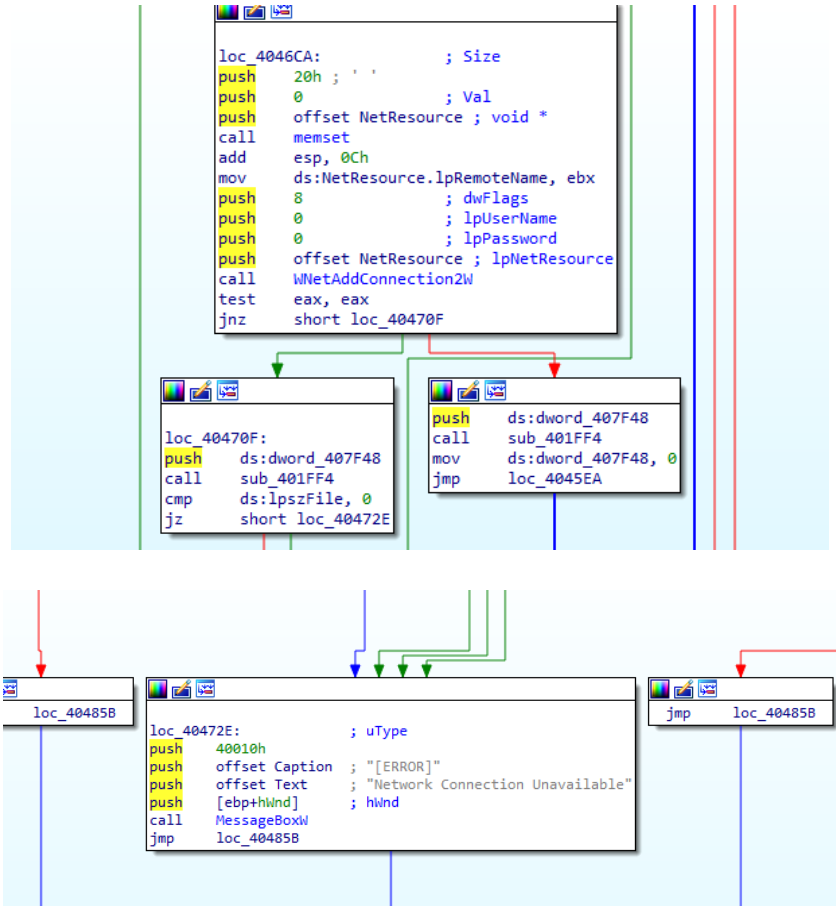
Based on the verdict of the searching of the files encrypted by LockBit Black is executed the push of the offset **"aFoundUFileS_0"** or **"aFoundUFileS":**



Here are the details of the logging in the trial_dec.log file:



In case the executable doesn't detect network connectivity the debugging tag **"[ERROR]"** is written**:**



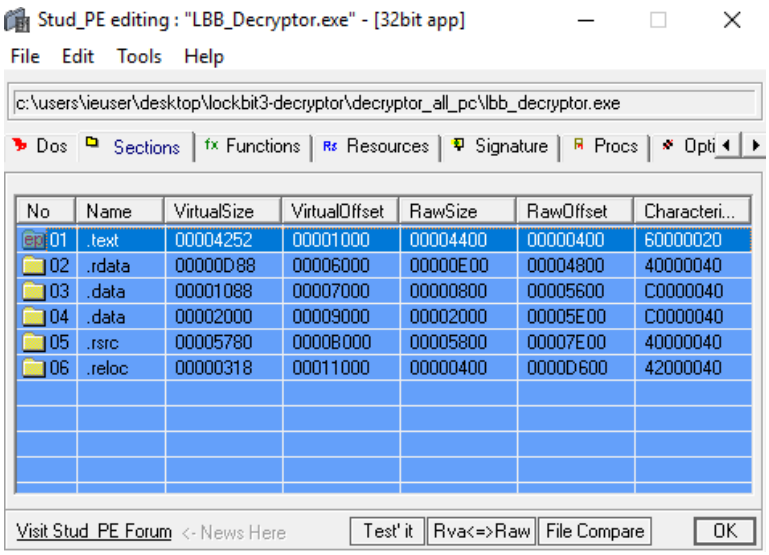
Following, the details of the most suspicious indicators of the executable in question:

indicator (39)	detail
The file references string(s)	type: blacklist, count: 30
The file execution privilege has been found	level: administrator
The file imports symbol(s)	type: blacklist, count: 29
The file references a string with a suspicious size	size: 1090 bytes
The time-stamp of a directory is suspicious	directory: debug, stamp: Thu Jul 14
The file references blacklist library(ies)	count: 1
The file references a group of API	type: windowing, count: 10
The file references a group of API	type: resource, count: 2
The file references a group of API	type: file, count: 44
The file references a group of API	type: execution, count: 31
The file references a group of API	type: console, count: 16
The file references a group of API	type: storage, count: 4
The file references a group of API	type: dynamic-library, count: 4
The file references a group of API	type: memory, count: 15
The file references a group of API	type: synchronization, count: 15
The file references a group of API	type: shell, count: 3
The file references a group of API	type: reckoning, count: 4
The file references a group of API	type: security, count: 8
The file references a group of API	type: cryptography, count: 8
The file references a group of API	type: registry, count: 8
The file references a group of API	type: network, count: 6
The file references a group of hint	type: file, count: 17
The file references a group of hint	type: function, count: 69
The file references a group of hint	type: size, count: 2
The file references a group of hint	type: format-string, count: 6
The file references a group of hint	type: utility, count: 3

Here's the Portable Executable sections, where it is highlight how the entropy of the .text section is **6.143**: this coefficient is quite low, so the analyzed decryptor doesn't present particular occultation techniques of the executable .text section.

Here are further details:

property	value	value	value
name	.text	.rdata	.data
md5	1BE6DF46D84967F7EA0AFD6...	1897CB665DDE48474BE633D...	05FD5450D3417D244B19AB2...
entropy	6.143	4.924	3.129
file-ratio (98.17%)	31.19 %	6.42 %	3.67 %
raw-address	0x00000400	0x00004800	0x00005600
raw-size (54784 bytes)	0x00004400 (17408 bytes)	0x00000E00 (3584 bytes)	0x00000800 (2048 bytes)
virtual-address	0x00401000	0x00406000	0x00407000
virtual-size (56058 bytes)	0x00004252 (16978 bytes)	0x00000D88 (3464 bytes)	0x00001088 (4232 bytes)
entry-point	**0x00004C72**	-	-
characteristics	0x60000020	0x40000040	0xC0000040
writable	-	-	**x**
executable	**x**	-	-
shareable	-	-	-
discardable	-	-	-
initialized-data	-	x	x
uninitialized-data	-	-	-
unreadable	-	-	-
self-modifying	-	-	-
virtualized	-	-	-
file	n/a	n/a	n/a



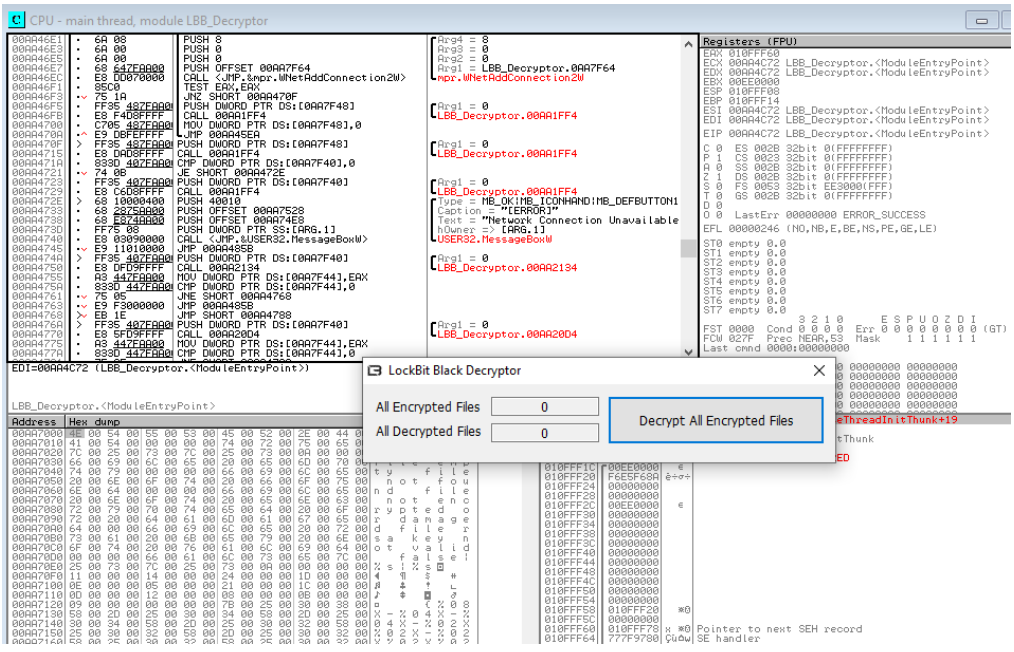
Here further details related to the imports performed and the number of functions associated to them:

library (9)	blacklist (1)	type (1)	functions (107)	description
user32.dll	-	implicit	14	Multi-User Windows USER API Client DLL
kernel32.dll	-	implicit	41	Windows NT BASE API Client DLL
comctl32.dll	-	implicit	1	Common Controls Library
shell32.dll	-	implicit	4	Windows Shell Common Dll
msvcrt.dll	-	implicit	11	Windows NT CRT DLL
advapi32.dll	-	implicit	7	Advanced Windows 32 Base API
ntdll.dll	-	implicit	19	NT Layer DLL
shlwapi.dll	-	implicit	8	Shell Light-weight Utility Library
mpr.dll	x	implicit	2	Multiple Provider Router DLL

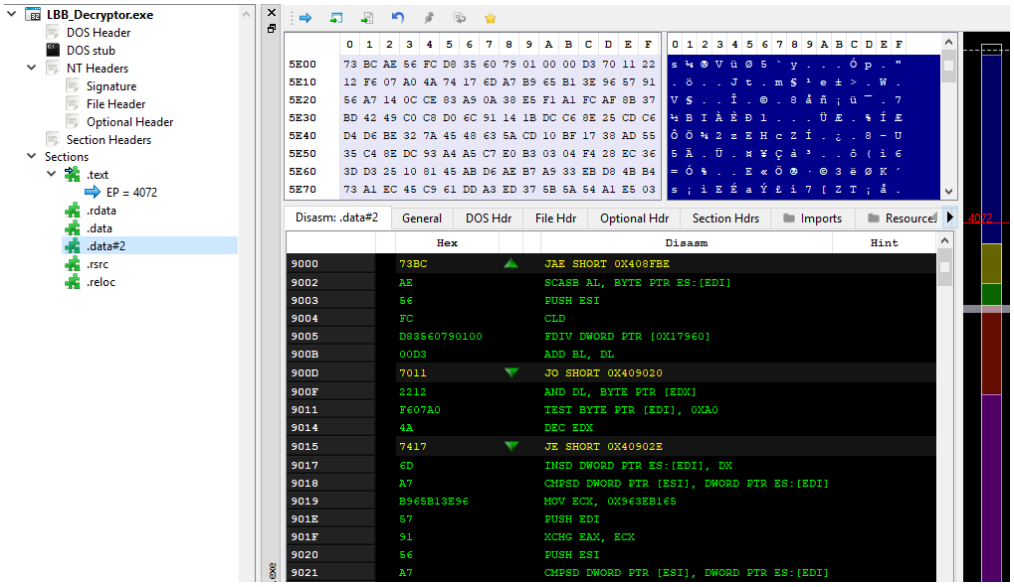
functions (107)	blacklist (29)	type (1)	ordinal (0)	library (9)
EnableWindow	-	implicit	-	user32.dll
DialogBoxParamW	-	implicit	-	user32.dll
SetDlgItemInt	-	implicit	-	user32.dll
SetSysColors	-	implicit	-	user32.dll
SetTimer	-	implicit	-	user32.dll
SetWindowPos	-	implicit	-	user32.dll
SetWindowTextW	-	implicit	-	user32.dll
SystemParametersInfoW	x	implicit	-	user32.dll
EndDialog	-	implicit	-	user32.dll
SendMessageW	-	implicit	-	user32.dll
MessageBoxW	-	implicit	-	user32.dll
LoadIconW	-	implicit	-	user32.dll
KillTimer	-	implicit	-	user32.dll
GetDlgItem	-	implicit	-	user32.dll
WriteFile	x	implicit	-	kernel32.dll
WriteConsoleW	-	implicit	-	kernel32.dll
WaitForSingleObject	-	implicit	-	kernel32.dll
WaitForMultipleObjects	-	implicit	-	kernel32.dll
Sleep	-	implicit	-	kernel32.dll
SetThreadPriority	-	implicit	-	kernel32.dll
SetFilePointerEx	-	implicit	-	kernel32.dll
CloseHandle	-	implicit	-	kernel32.dll
CreateFileW	-	implicit	-	kernel32.dll
CreateIoCompletionPort	-	implicit	-	kernel32.dll
CreateThread	-	implicit	-	kernel32.dll
DeleteFileW	x	implicit	-	kernel32.dll
ExitProcess	-	implicit	-	kernel32.dll
FindClose	-	implicit	-	kernel32.dll
FindFirstFileExW	x	implicit	-	kernel32.dll
FindNextFileW	x	implicit	-	kernel32.dll
FlushConsoleInputBuffer	x	implicit	-	kernel32.dll

functions (107)	blacklist (29)	type (1)	ordinal (0)	library (9)
wcscat	-	implicit	-	msvcrt.dll
wcscpy	-	implicit	-	msvcrt.dll
MD5Update	x	implicit	-	advapi32.dll
MD5Init	x	implicit	-	advapi32.dll
MD5Final	x	implicit	-	advapi32.dll
ConvertSidToStringSidW	-	implicit	-	advapi32.dll
RegDeleteKeyW	x	implicit	-	advapi32.dll
RegSetValueExW	x	implicit	-	advapi32.dll
RegCreateKeyExW	-	implicit	-	advapi32.dll
RtlDeleteCriticalSection	-	implicit	-	ntdll.dll
RtlDestroyHeap	-	implicit	-	ntdll.dll
RtlCreateHeap	-	implicit	-	ntdll.dll
RtlFreeHeap	-	implicit	-	ntdll.dll
RtlInitializeCriticalSection	-	implicit	-	ntdll.dll
RtlLeaveCriticalSection	-	implicit	-	ntdll.dll
RtlReAllocateHeap	-	implicit	-	ntdll.dll
NtClose	-	implicit	-	ntdll.dll
RtlAllocateHeap	-	implicit	-	ntdll.dll
RtlAdjustPrivilege	x	implicit	-	ntdll.dll
NtTerminateThread	-	implicit	-	ntdll.dll
NtSetInformationThread	x	implicit	-	ntdll.dll
NtSetInformationProcess	x	implicit	-	ntdll.dll
NtQuerySystemInformation	x	implicit	-	ntdll.dll
NtQueryInformationToken	-	implicit	-	ntdll.dll
NtOpenProcessToken	x	implicit	-	ntdll.dll
NtOpenProcess	x	implicit	-	ntdll.dll
NtDuplicateToken	-	implicit	-	ntdll.dll
RtlEnterCriticalSection	-	implicit	-	ntdll.dll
PathFindFileNameW	x	implicit	-	shlwapi.dll
PathIsDirectoryEmptyW	x	implicit	-	shlwapi.dll
PathFindExtensionW	x	implicit	-	shlwapi.dll

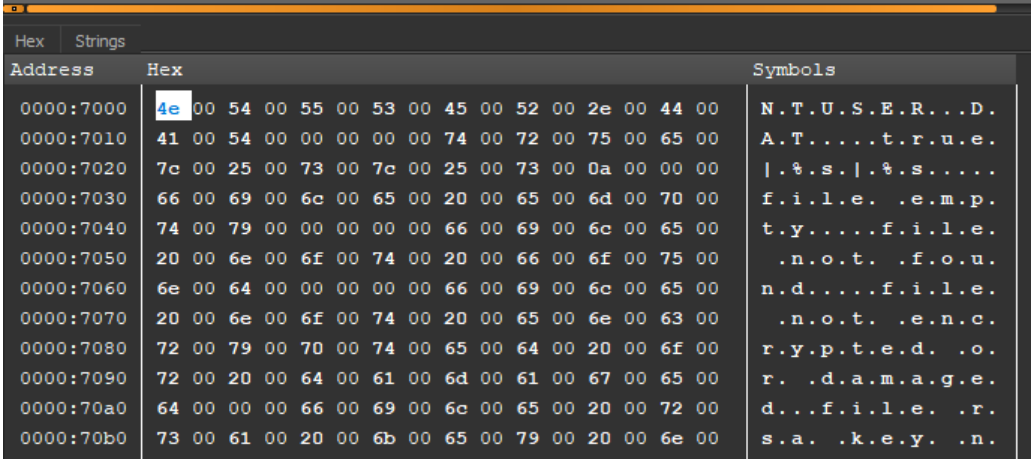
Here's how the decryptor is in a debugging instance:



Curiously, the executable has two .data sections. In the second section there are details related to strings used as output during the "compare" instructions as verdicts of the decryption threads:



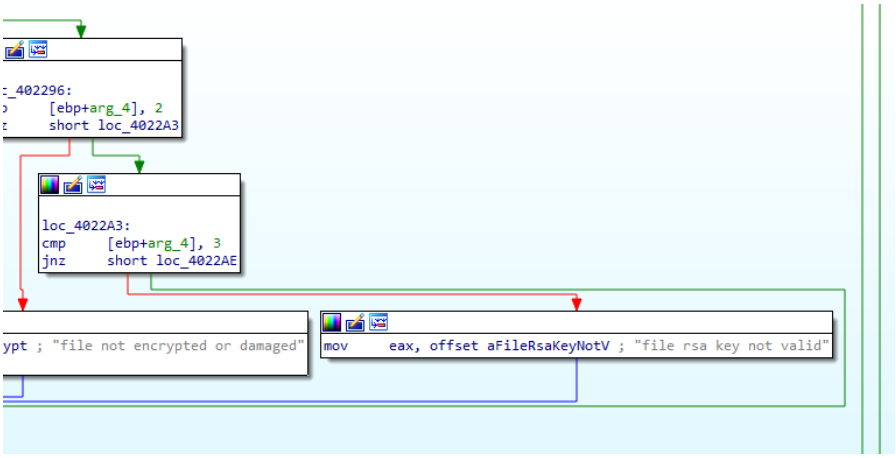
	Name	VirtualSize	irtualAddres	zeOfRawDa	nterToRawD	terToReloca	erToLinenur	berOfReloca	erOfLinenu
0	.text	00004252	00001000	00004400	00000400	00000000	00000000	0000	0000
1	.rdata	00000d88	00006000	00000e00	00004800	00000000	00000000	0000	0000
2	.data	00001088	00007000	00000800	00005600	00000000	00000000	0000	0000
3	.data	00002000	00009000	00002000	00005e00	00000000	00000000	0000	0000
4	.rsrc	00005780	0000b000	00005800	00007e00	00000000	00000000	0000	0000
5	.reloc	00000318	00011000	00000400	0000d600	00000000	00000000	0000	0000

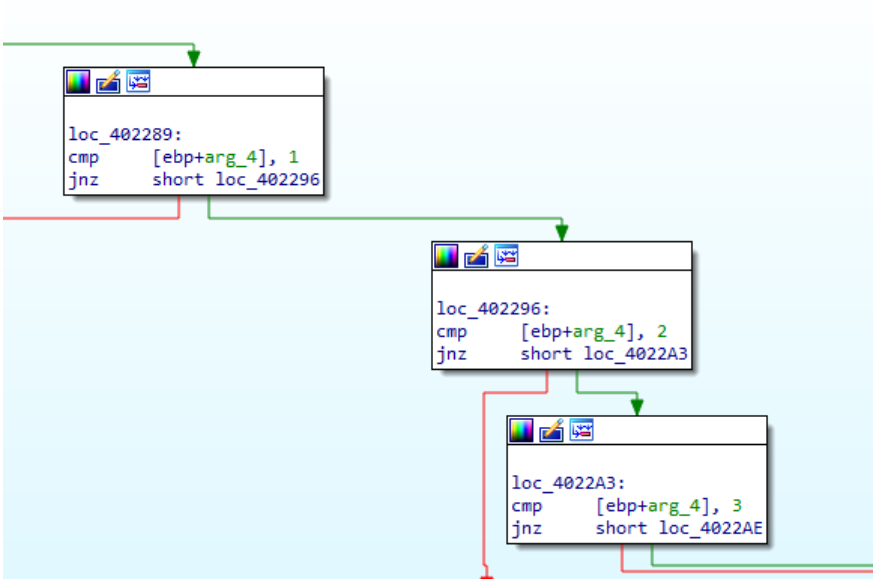


	Address	Size	Type	String
1	7000	0000000a	U	NTUSER.DAT
2	7030	0000000a	U	file empty
3	7048	0000000e	U	file not found
4	7068	0000001d	U	file not encrypted or damaged
5	70a4	00000016	U	file rsa key not valid
6	7128	00000032	U	{%08X-%04X-%04X-%02X%02X-%02X%02X%02X%02X%02X%02X}
7	7190	0000000d	U	%s.README.txt
8	71c4	0000000c	U	\DefaultIcon
9	71e0	0000000c	U	advapi32.dll

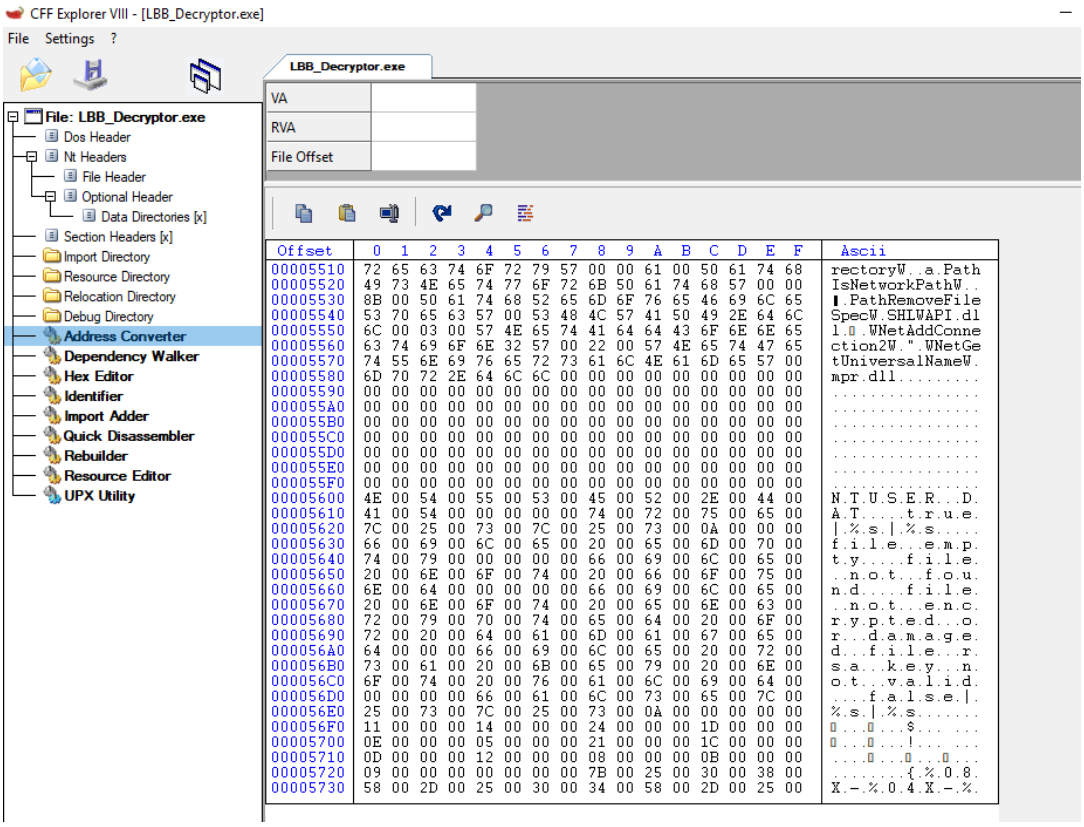
```
.data:004070A4 aFileRsaKeyNotV:                        ; DATA XREF: sub_402240+69↑o
.data:004070A4                 text "UTF-16LE", 'file rsa key not valid',0
.data:004070D2                 align 4
```

After a series of checks (*"compare"*) and conceptual *"if"*, we arrive at the result "file rsa key not found":





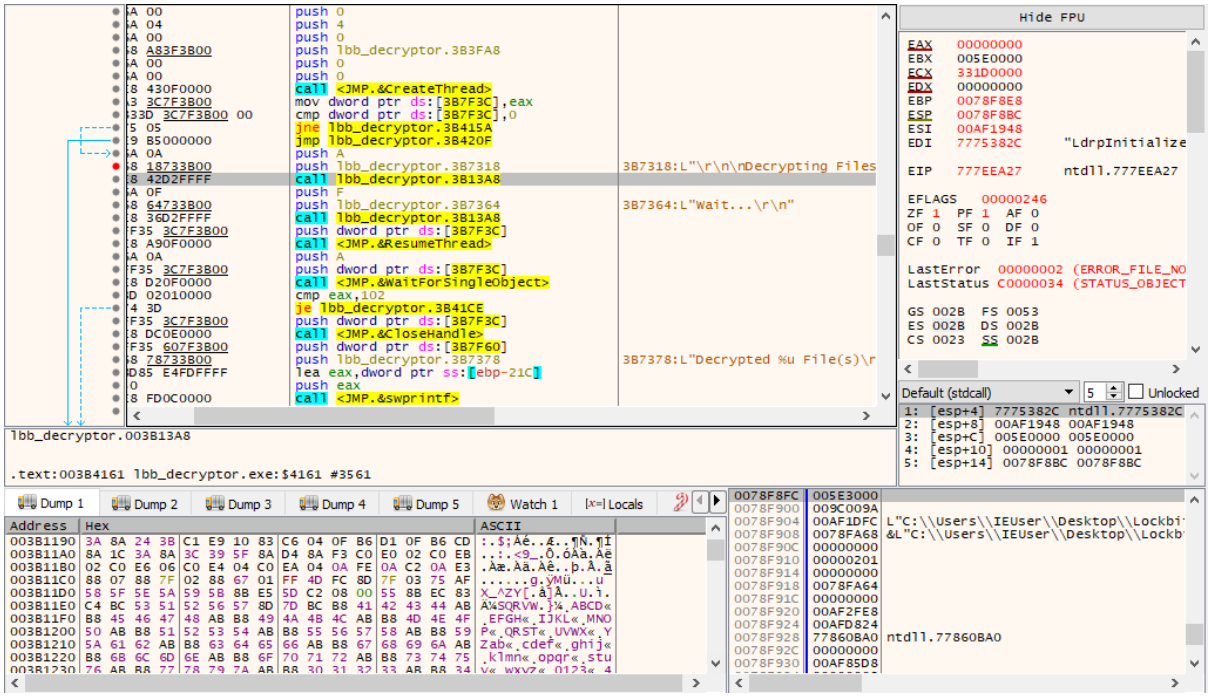
Here's the details of an extract of the hexadecimal code of the analyzed executable:



LBB_Decryptor.exe

Member	Offset	Size	Value	Meaning
Magic	00000098	Word	010B	PE32
MajorLinkerVersion	0000009A	Byte	0E	
MinorLinkerVersion	0000009B	Byte	0C	
SizeOfCode	0000009C	Dword	00004400	
SizeOfInitializedData	000000A0	Dword	00009C00	
SizeOfUninitializedData	000000A4	Dword	00000000	
AddressOfEntryPoint	000000A8	Dword	00004C72	.text
BaseOfCode	000000AC	Dword	00001000	
BaseOfData	000000B0	Dword	00006000	
ImageBase	000000B4	Dword	00400000	
SectionAlignment	000000B8	Dword	00001000	
FileAlignment	000000BC	Dword	00000200	
MajorOperatingSystemVers...	000000C0	Word	0005	
MinorOperatingSystemVers...	000000C2	Word	0001	
MajorImageVersion	000000C4	Word	0000	
MinorImageVersion	000000C6	Word	0000	
MajorSubsystemVersion	000000C8	Word	0005	
MinorSubsystemVersion	000000CA	Word	0001	
Win32VersionValue	000000CC	Dword	00000000	
SizeOfImage	000000D0	Dword	00012000	
SizeOfHeaders	000000D4	Dword	00000400	
CheckSum	000000D8	Dword	0000FFC3	

Following, the details related to the phase of the execution of the decryption thread where is possible to notice references to the management of registry keys for their disinfection:




```
int32_t SetFileAttributesW = 0x68dc;

void fun_40513e(struct s0* a1, struct s0* a2, struct s0* a3) {
    goto SetFileAttributesW;
}

int32_t DeleteFileW = 0x66ba;

struct s0* fun_405090(struct s0* a1, struct s0* a2, struct s0* a3, struct s0* a4, struct s0* a5, struct s0* a6, struct s0* a7) {
    goto DeleteFileW;
}

int32_t SHGetSpecialFolderPathW = 0x69da;

void fun_405180(struct s0* a1, struct s0* a2, struct s0* a3, struct s0* a4) {
    goto SHGetSpecialFolderPathW;
}

void fun_401574(struct s0* ecx, struct s0* a2, struct s0* a3, struct s0* a4, struct s0* a5, struct s0* a6, struct s0* a7, struct s0* a8, struct
    int32_t eax19;
    struct s0* ebx20;

    if (!a2) {
        addr_40159d_2:
        return;
    } else {
        eax19 = 0;
        ebx20 = a2;
        while (*reinterpret_cast<struct s0**>(reinterpret_cast<unsigned char>(ebx20) + eax19 * 2)) {
```

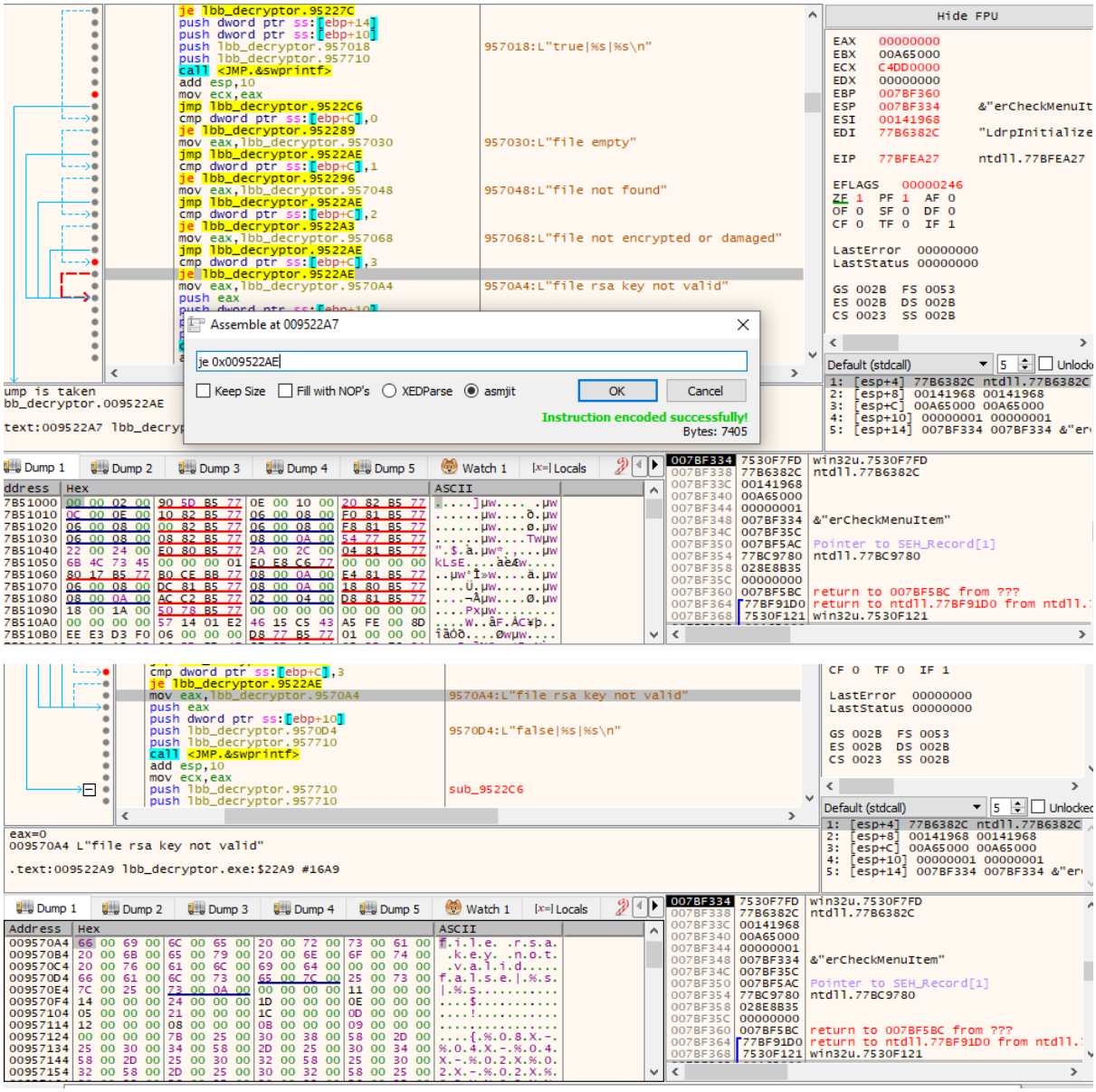
```
int32_t PathFindExtensionW = 0x6cc4;

struct s0* fun_40522e(struct s0* a1, struct s0* a2, struct s0* a3, struct s0* a4, struct s0* a5, struct s0* a6) {
    goto PathFindExtensionW;
}

int32_t GetModuleHandleW = 0x67b8;

struct s0* fun_4050e4(struct s0* a1, struct s0* a2, struct s0* a3, struct s0* a4, struct s0* a5, struct s0* a6, struct s0* a7, struct s0* a8, st
    goto GetModuleHandleW;
}

int32_t GetProcAddress = 0x67cc;

struct s0* fun_4050ea(struct s0* a1, struct s0* a2, struct s0* a3, struct s0* a4, struct s0* a5, struct s0* a6, struct s0* a7, struct s0* a8, st
    goto GetProcAddress;
}

int32_t RegSetValueExW = 0x6adc;

void fun_4051aa(int32_t a1, int32_t a2, int32_t a3, int32_t a4, int32_t a5, int32_t a6, struct s0* a7, struct s0* a8, struct s0* a9, struct s0
    goto RegSetValueExW;
}

int32_t SystemParametersInfoW = 0x6650;

void fun_405072(int32_t a1, int32_t a2, int32_t a3, uint32_t a4, struct s0* a5, struct s0* a6, struct s0* a7, struct s0* a8, struct s0* a9, st
    goto SystemParametersInfoW;
}

int32_t SHChangeNotify = 0x69c8;

struct s0* fun_40517a(int32_t a1, int32_t a2, int32_t a3, int32_t a4, int32_t a5, int32_t a6, int32_t a7, uint32_t a8, struct s0* a9, struct
    goto SHChangeNotify;
}
```

```
int32_t NtQuerySystemInformation = 0x6b5a;

int32_t fun_4051ce() {
    goto NtQuerySystemInformation;
}

int32_t RtlReAllocateHeap = 0x6c86;

void fun_40521c() {
    goto RtlReAllocateHeap;
}

int32_t RtlFreeHeap = 0x6c3e;

void fun_40520a() {
    goto RtlFreeHeap;
}

int32_t RtlAdjustPrivilege = 0x6bbe;

int32_t fun_4051e6(int32_t a1, int32_t a2, int32_t a3, void* a4) {
    goto RtlAdjustPrivilege;
}

int32_t RegCreateKeyExW = 0x6aba;

int32_t fun_40519e(struct s0* a1, struct s0* a2, struct s0* a3, struct s0* a4, struct s0* a5, struct s0* a6, struct s0* a7, struct s0* a8, stru
    goto RegCreateKeyExW;
}
```

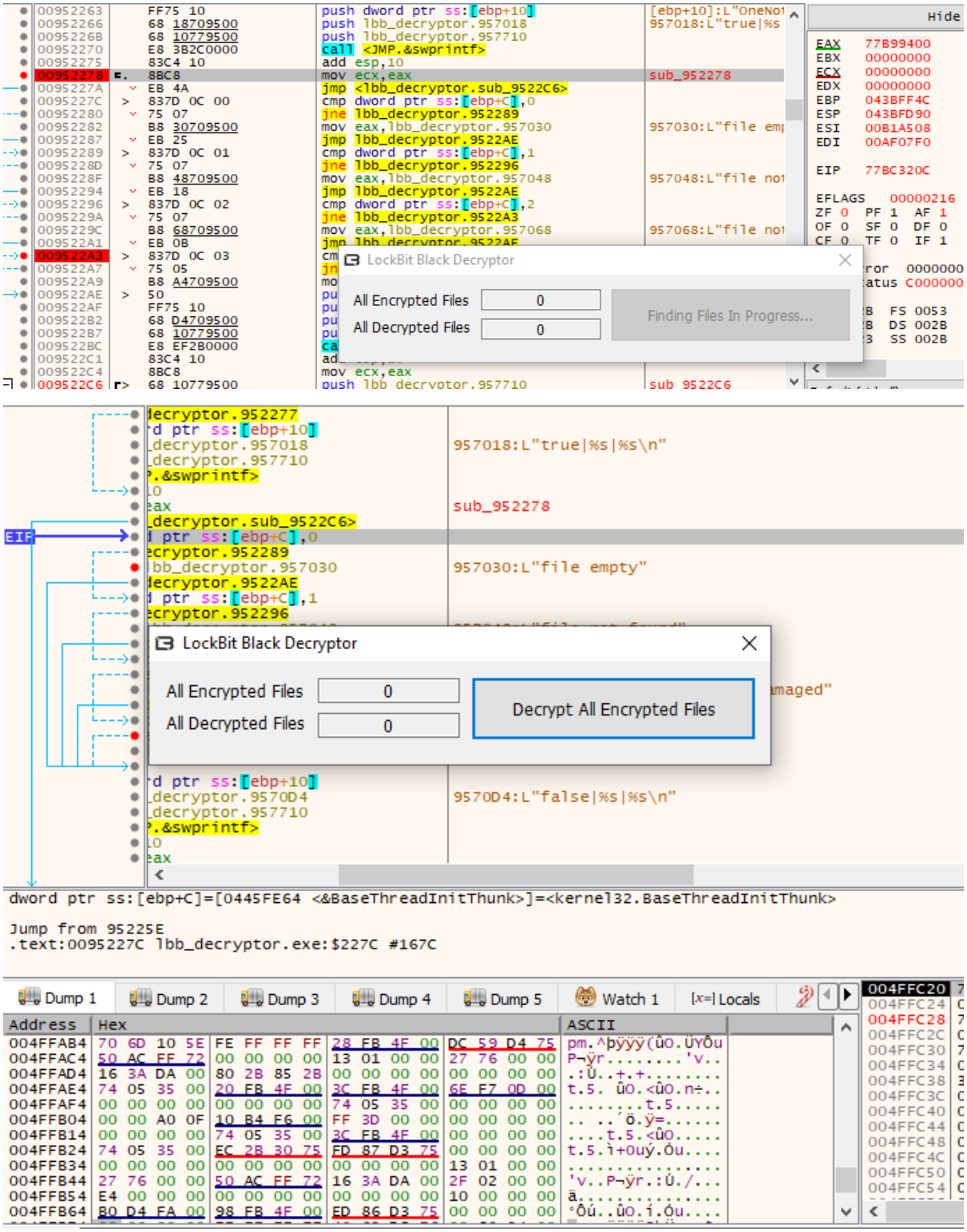
```
if (v4) {
  v5 = reinterpret_cast<struct s0*>(reinterpret_cast<int32_t>(ebp2) + 0xfffffde8);
  fun_405180(0, v5, 35, 0);
  fun_401574(ecx, reinterpret_cast<int32_t>(ebp2) + 0xfffffde8, 0, v5, 35, 0, edi6, esi7, edx8, ecx, ebx9, v10, v11, v12, v13, v14, v15, v16)
  fun_404eb6(reinterpret_cast<int32_t>(ebp2) + 0xfffffde8, &v4->f2, 0, v5, 35, 0, edi6, esi7, edx8, ecx, ebx9, v17, v18, v19, v20, v21, v22,
  fun_404eb6(reinterpret_cast<int32_t>(ebp2) + 0xfffffde8, ".", 0, v5, 35, 0, edi6, esi7, edx8, ecx, ebx9, v24, v25, v26, v27, v28, v29, v30);
  v31 = reinterpret_cast<struct s0*>(reinterpret_cast<int32_t>(ebp2) + 0xfffffde8);
  fun_405090(v31, 0, v5, 35, 0, edi6, esi7);
  v32 = reinterpret_cast<struct s0*>(reinterpret_cast<int32_t>(ebp2) + 0xfffffde8);
  eax33 = fun_40522e(v32, v31, 0, v5, 35, 0);
  fun_404ebc(eax33, ".", v32, v31, 0, v5, 35, 0, edi6);
  v34 = reinterpret_cast<struct s0*>(reinterpret_cast<int32_t>(ebp2) + 0xfffffde8);
  fun_405090(v34, v32, v31, 0, v5, 35, 0);
  fun_404ebc(reinterpret_cast<int32_t>(ebp2) + 0xfffffde8, &v4->f2, v34, v32, v31, 0, v5, 35, 0);
  fun_404eb6(reinterpret_cast<int32_t>(ebp2) + 0xfffffde8, "\\", v34, v32, v31, 0, v5, 35, 0, edi6, esi7, edx8, ecx, ebx9, v35, v36, v37, v38)
  eax42 = fun_4050e4("a", v34, v32, v31, 0, v5, 35, 0, edi6, esi7, edx8, ecx, ebx9, v39, v40, v41);
  eax46 = fun_4050ea(eax42, "RegDeleteKeyExW", "a", v34, v32, v31, 0, v5, 35, 0, edi6, esi7, edx8, ecx, ebx9, v43, v44, v45);
  g407f10 = eax46;
  zf47 = g407f10 == 0;
  if (zf47) {
    v48 = reinterpret_cast<struct s0*>(reinterpret_cast<int32_t>(ebp2) + 0xfffffde8);
    v49 = reinterpret_cast<struct s0*>(0x80000000);
    fun_4051a4(0x80000000, v48, eax42, "RegDeleteKeyExW", "a", v34, v32, v31, 0, v5, 35, 0, edi6, esi7, edx8, ecx);
    v50 = reinterpret_cast<struct s0*>(&v4->f2);
    fun_4051a4(0x80000000, v50, 0x80000000, v48, eax42, "RegDeleteKeyExW", "a", v34, v32, v31, 0, v5, 35, 0, edi6, esi7);
    v51 = v4;
    v52 = reinterpret_cast<struct s0*>(0x80000000);
    fun_4051a4(0x80000000, v51, 0x80000000, v50, 0x80000000, v48, eax42, "RegDeleteKeyExW", "a", v34, v32, v31, 0, v5, 35, 0);
  } else {
    v48 = reinterpret_cast<struct s0*>(0);
    v49 = reinterpret_cast<struct s0*>(0x100);
    v50 = reinterpret_cast<struct s0*>(reinterpret_cast<int32_t>(ebp2) + 0xfffffde8);
    g407f10(0x80000000, v50, 0x100, 0, eax42, "RegDeleteKeyExW", "a", v34, v32, v31, 0, v5, 35, 0);
    v51 = reinterpret_cast<struct s0*>(0);
    v52 = reinterpret_cast<struct s0*>(0x100);
    v53 = &v4->f2;
    g407f10(0x80000000, v53, 0x100, 0, 0x80000000, v50, 0x100, 0, eax42, "RegDeleteKeyExW", "a", v34, v32, v31, 0, v5, 35, 0);
    g407f10(0x80000000, v4, 0x100, 0, 0x80000000, v53, 0x100, 0, 0x80000000, v50, 0x100, 0, eax42, "RegDeleteKeyExW", "a", v34, v32, v31,
  }
```

After the execution of the subroutines of the management of the files attributes, it is executed a series of compare instructions.

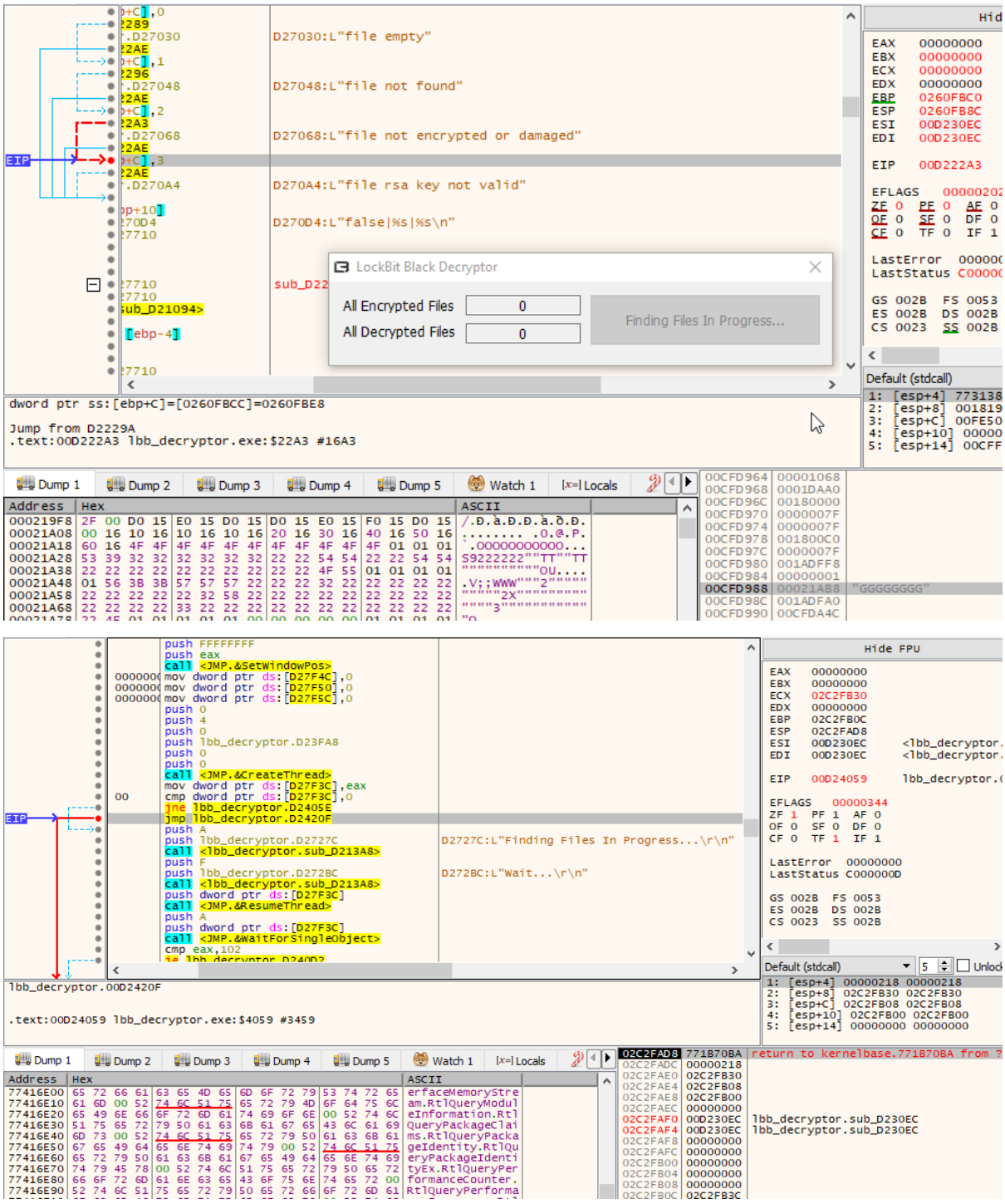
In the following case it doesn't succeed the phase of the decryption of files (obtained through enumeration), in fact the verdict is "false". In case the results of the operations are true (so succeded) a positive verdict of the execution is written in the log file trial_dec.log (then called from the filepath trailing removing function PathRemoveFileSpecW):



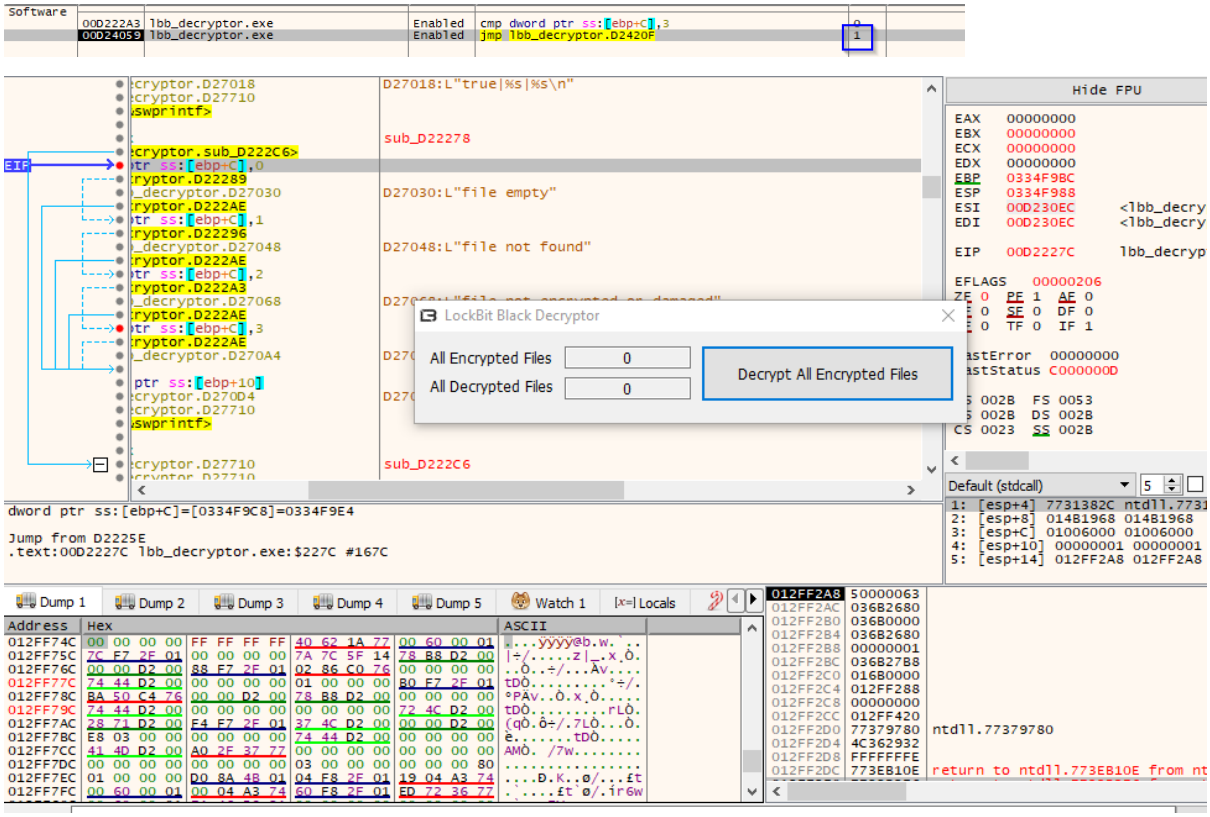
Following some details of the dumping of the EBP register, which is used to perform the compare instructions and execute the operation of switching through the various output options:



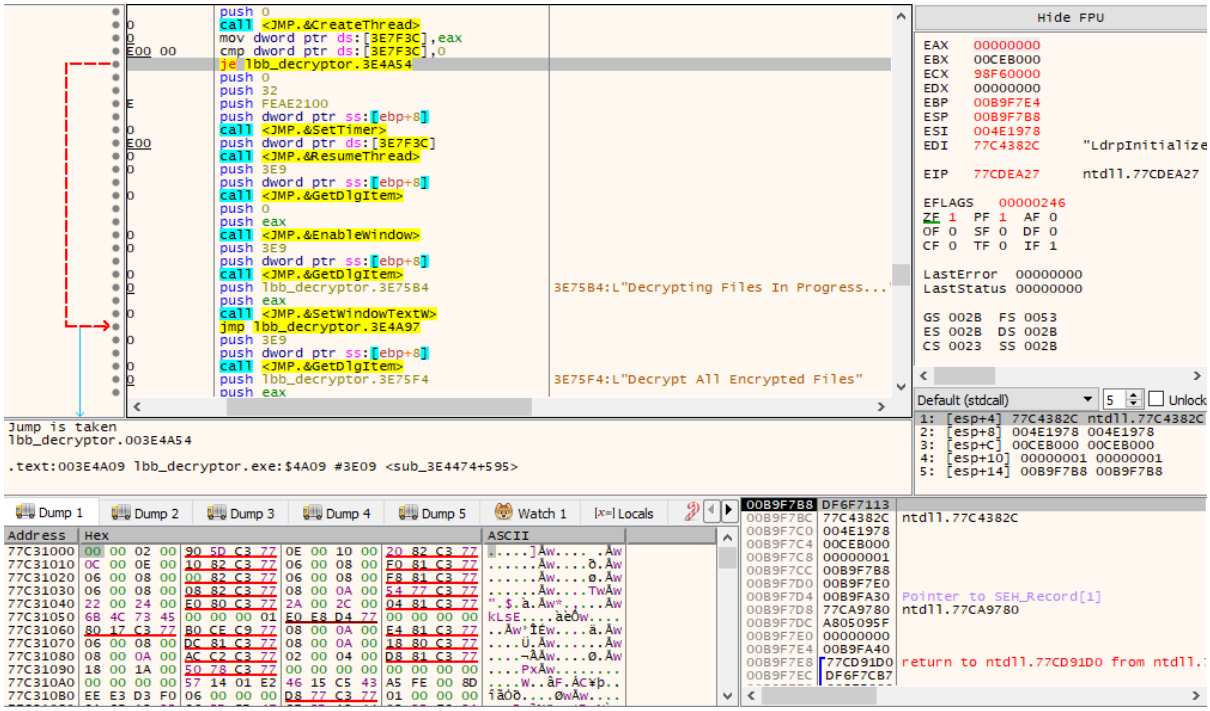
Here's a detail of debugging with the EIP pointed before the verdict **"RSA key not valid"**, so it is simulated the read of the content of the registers (so the data related to the execution) during the decryption and logging phase:

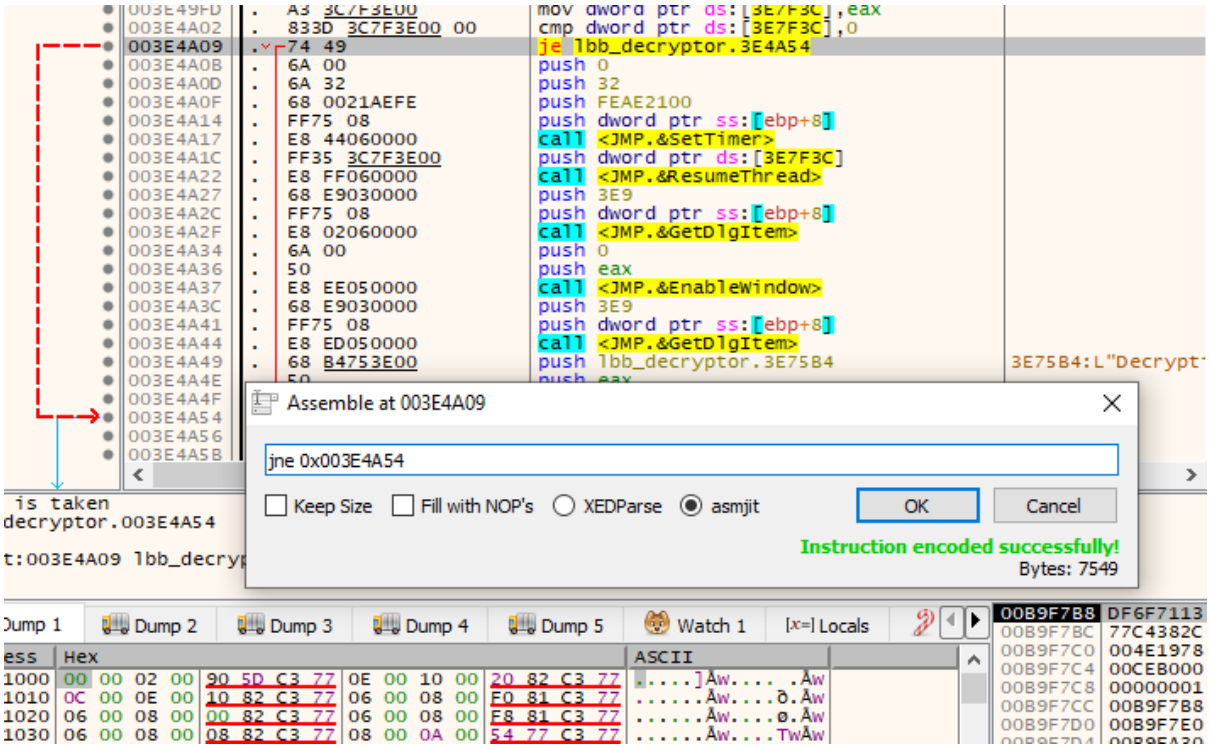


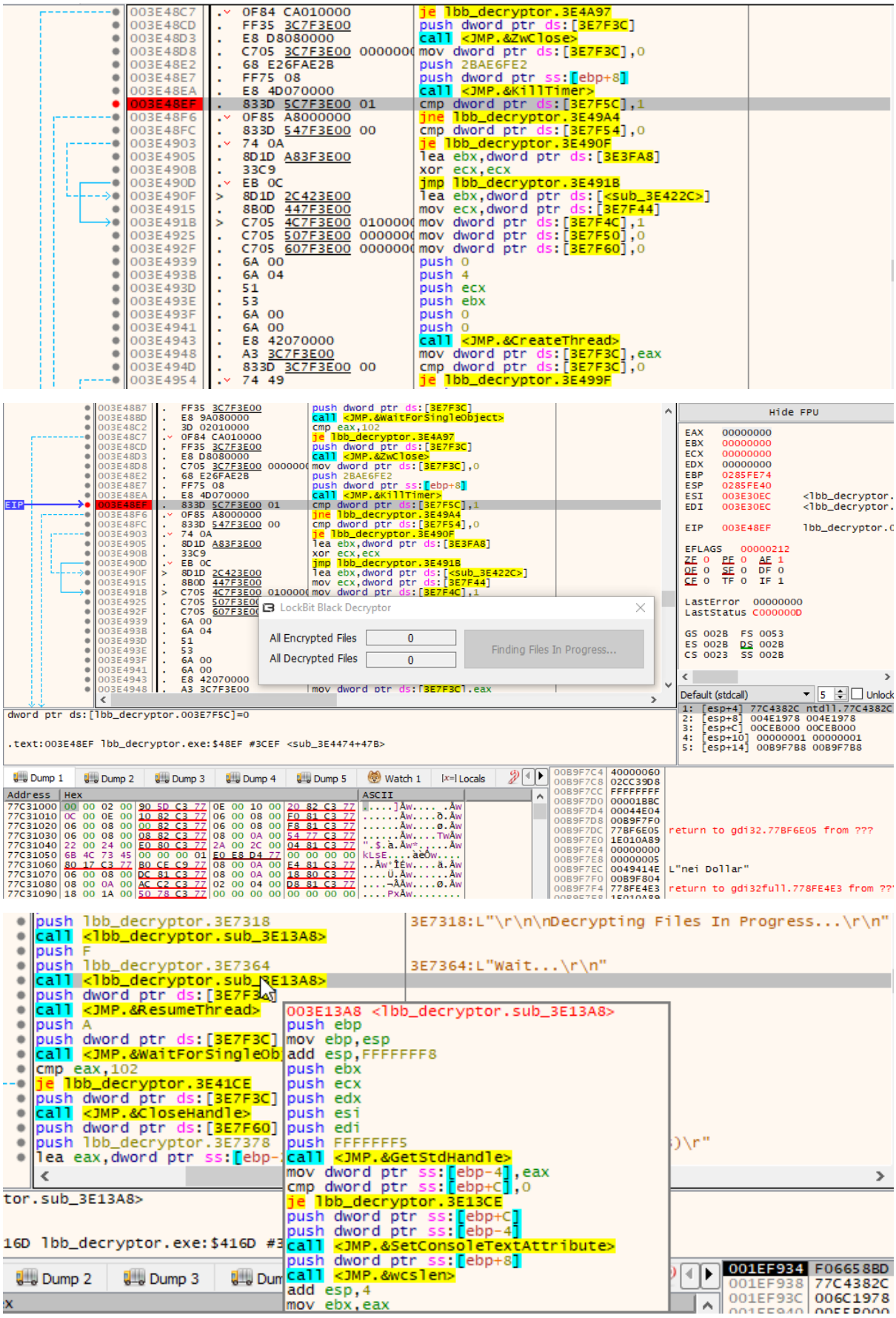
Following the evidence of breakpoint hitting where you can see that during the phase of debugging it was intercepted the breakpoint previously set:



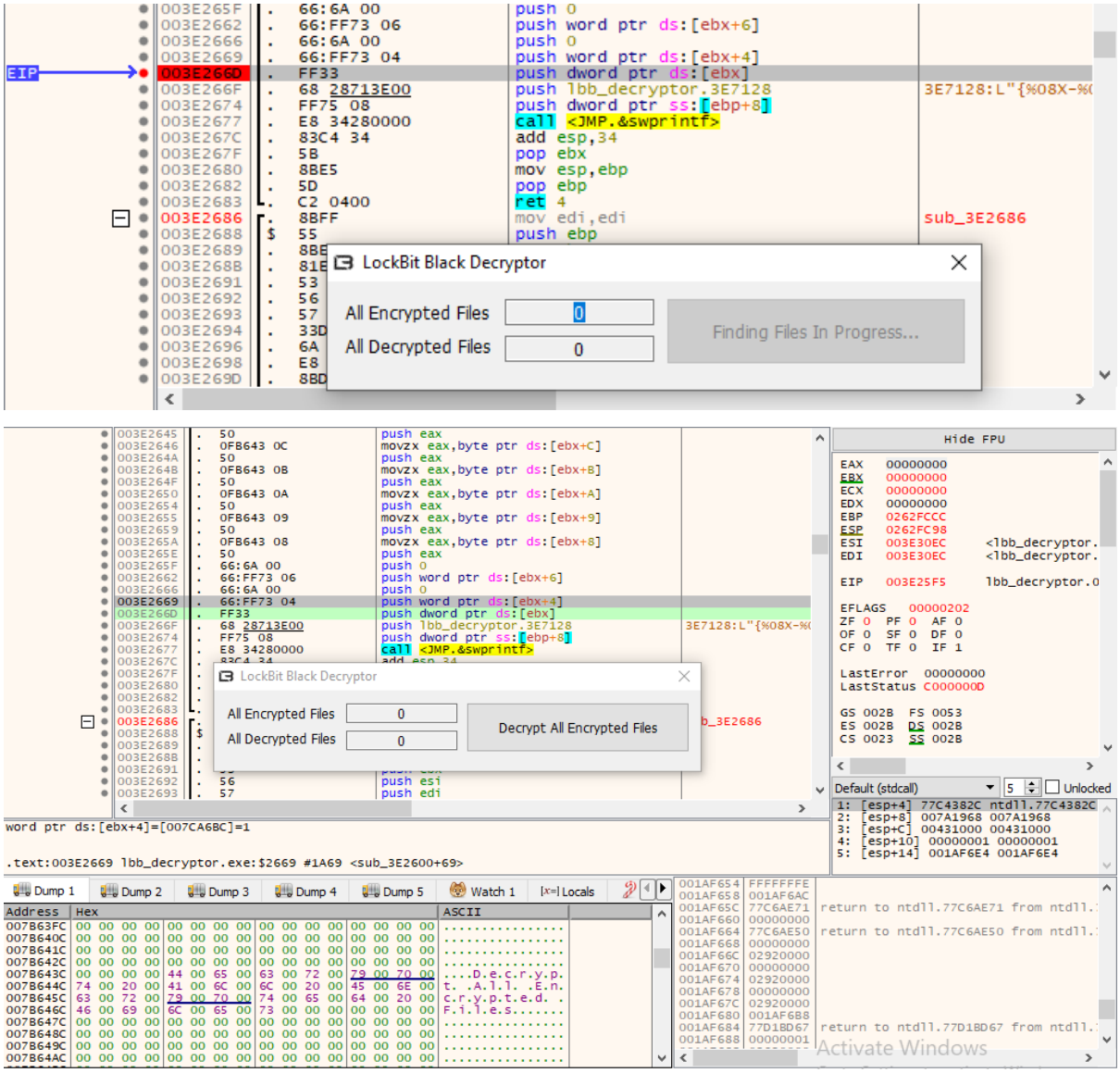
Following, the details of an attempt to modify a **JE** assembly instruction in the context of the creation of the bulk decryption thread to modify the execution flow after the creation of it:



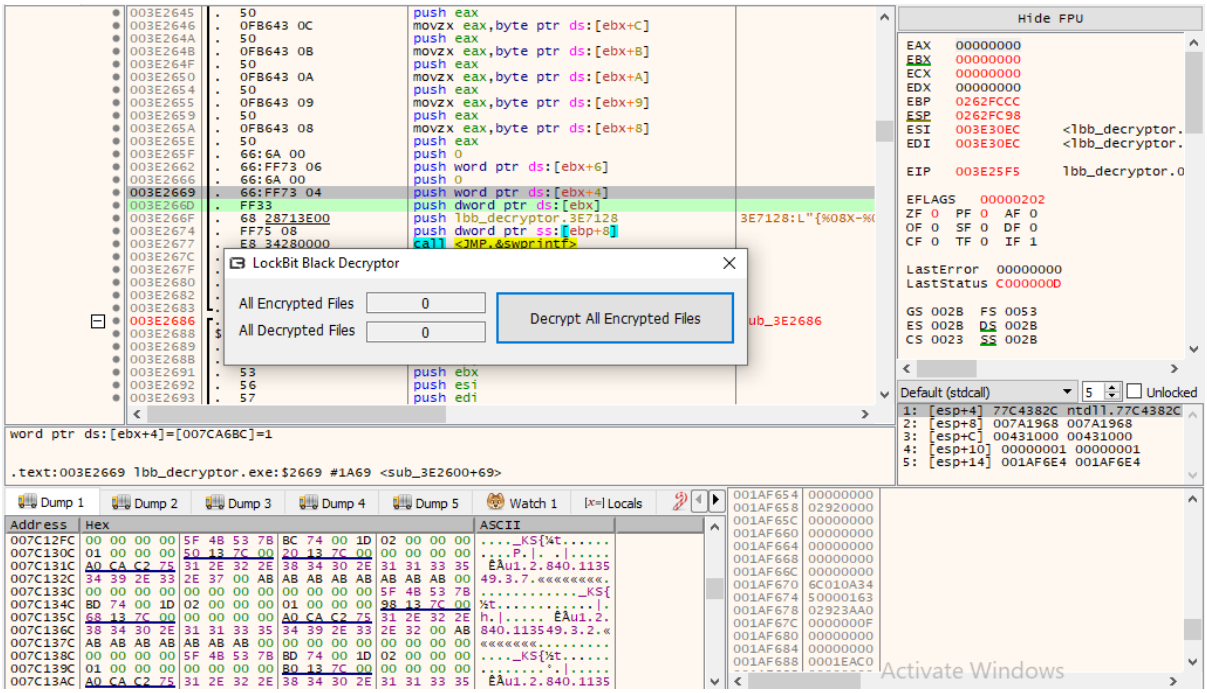

Assemble at 003E4A09
jne 0x003E4A54
Keep Size
Fill with NOP's
XEDParse
asmjit
OK
Cancel
Instruction encoded successfully!
Bytes: 7549


LockBit Black Decryptor
All Encrypted Files
0
All Decrypted Files
0
Finding Files In Progress...
Hide FPU
Dump 1
Dump 2
Dump 3
Dump 4
Dump 5
Watch 1
Locals
Address
Hex
ASCII
push lbb_decryptor.3E7318
call <lbb_decryptor.sub_3E13A8>
push F
push lbb_decryptor.3E7364
call <lbb_decryptor.sub_3E13A8>
push dword ptr ds:[3E7F3C]
call <JMP.&ResumeThread>
push A
push dword ptr ds:[3E7F3C]
call <JMP.&WaitForSingleObject>
cmp eax,102
je lbb_decryptor.3E41CE
push dword ptr ds:[3E7F3C]
call <JMP.&CloseHandle>
push dword ptr ds:[3E7F60]
push lbb_decryptor.3E7378
lea eax,dword ptr ss:[ebp-
3E7318:L"\r\n\nDecrypting Files In Progress...\r\n"
3E7364:L"Wait...\r\n"
003E13A8 <lbb_decryptor.sub_3E13A8>
push ebp
mov ebp,esp
add esp,FFFFFFF8
push ebx
push ecx
push edx
push esi
push edi
push FFFFFFF5
call <JMP.&GetStdHandle>
mov dword ptr ss:[ebp-4],eax
cmp dword ptr ss:[ebp+C],0
je lbb_decryptor.3E13CE
push dword ptr ss:[ebp+C]
push dword ptr ss:[ebp-4]
call <JMP.&SetConsoleTextAttribute>
push dword ptr ss:[ebp+8]
call <JMP.&wcslen>
add esp,4
mov ebx,eax

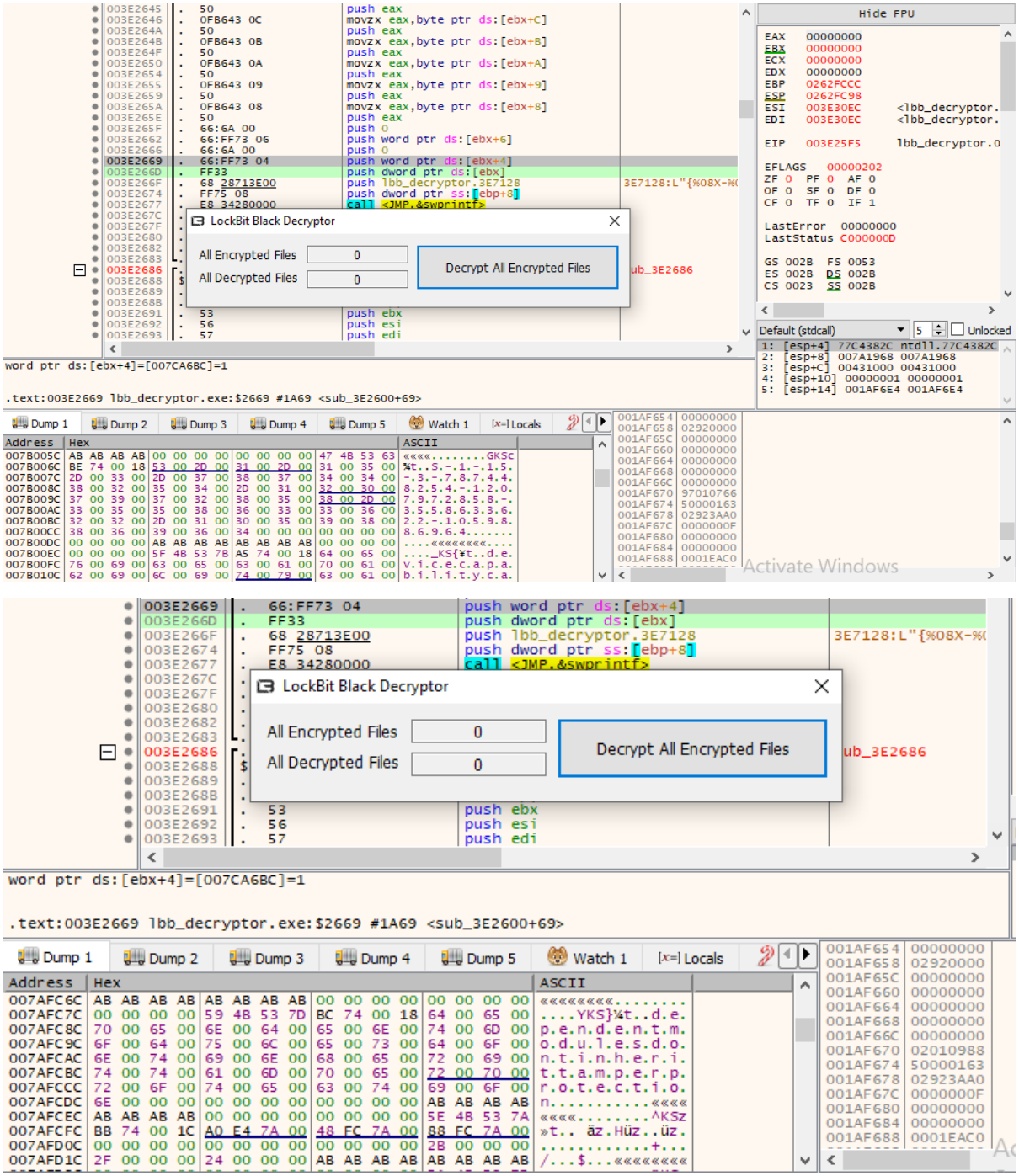
Here are, instead, details related to a session of breakpoint analysis when the buffer is filled with the ID during the MD5 digest:

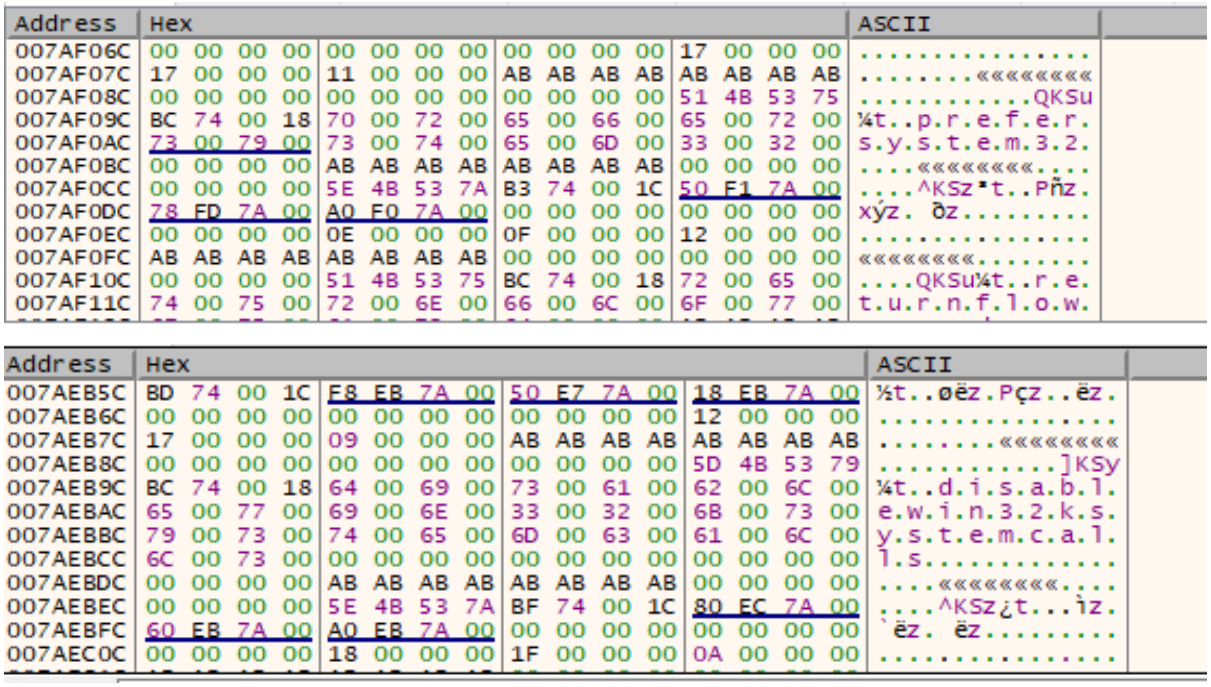


By performing a dump of the EBX register it is possible to see some embedded numeric patterns which, with probability, are related to the digesting operation:



It is possible to highlight the presence of a security identifier. The decryptor tool performs infact also OS settings and information gathering. Those evidences show, however, the trend to customize those decryptor tools for the victims.

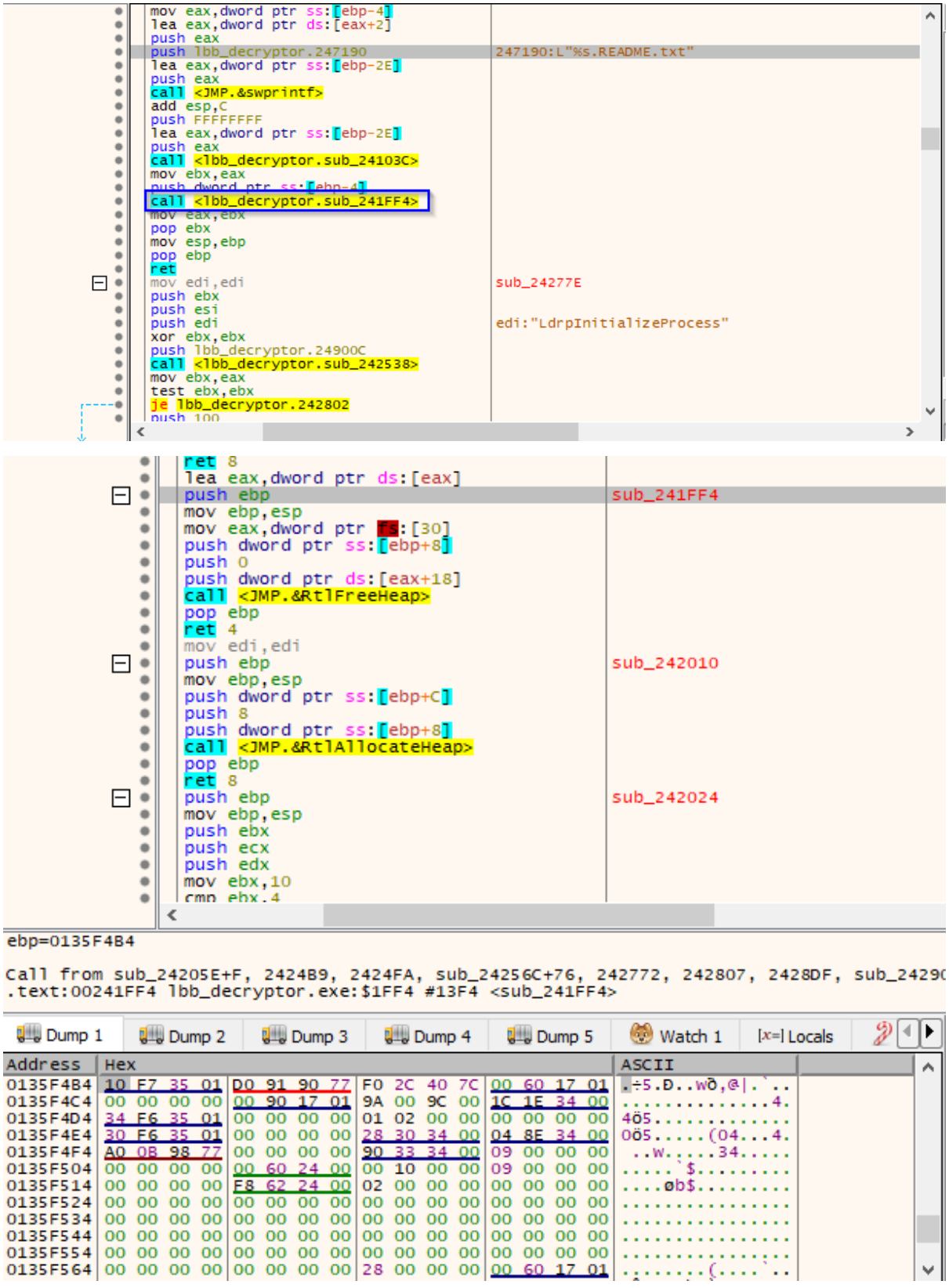

LockBit Black Decryptor
All Encrypted Files
0
All Decrypted Files
0
Decrypt All Encrypted Files
Hide FPU
word ptr ds:[ebx+4]=[007CA6BC]=1
.text:003E2669 lbb_decryptor.exe:$2669 #1A69 <sub_3E2600+69>
Dump 1
Dump 2
Dump 3
Dump 4
Dump 5
Watch 1
[x=] Locals



Address	Hex																ASCII
007AF06C	00	00	00	00	00	00	00	00	00	00	00	00	17	00	00	00	....
007AF07C	17	00	00	00	11	00	00	00	AB	AB	AB	AB	AB	AB	AB	AB	........««««««««
007AF08C	00	00	00	00	00	00	00	00	00	00	00	00	51	4B	53	75	............QKSu
007AF09C	BC	74	00	18	70	00	72	00	65	00	66	00	65	00	72	00	¼t..p.r.e.f.e.r.
007AF0AC	73	00	79	00	73	00	74	00	65	00	6D	00	33	00	32	00	s.y.s.t.e.m.3.2.
007AF0BC	00	00	00	00	AB	AB	AB	AB	AB	AB	AB	AB	00	00	00	00	....«««««««....
007AF0CC	00	00	00	00	5E	4B	53	7A	B3	74	00	1C	50	F1	7A	00	....^KSz³t..Pñz.
007AF0DC	78	FD	7A	00	A0	F0	7A	00	00	00	00	00	00	00	00	00	xýz. ðz.........
007AF0EC	00	00	00	00	0E	00	00	00	0F	00	00	00	12	00	00	00	....
007AF0FC	AB	AB	AB	AB	AB	AB	AB	AB	00	00	00	00	00	00	00	00	««««««««........
007AF10C	00	00	00	00	51	4B	53	75	BC	74	00	18	72	00	65	00	....QKSu¼t..r.e.
007AF11C	74	00	75	00	72	00	6E	00	66	00	6C	00	6F	00	77	00	t.u.r.n.f.l.o.w.

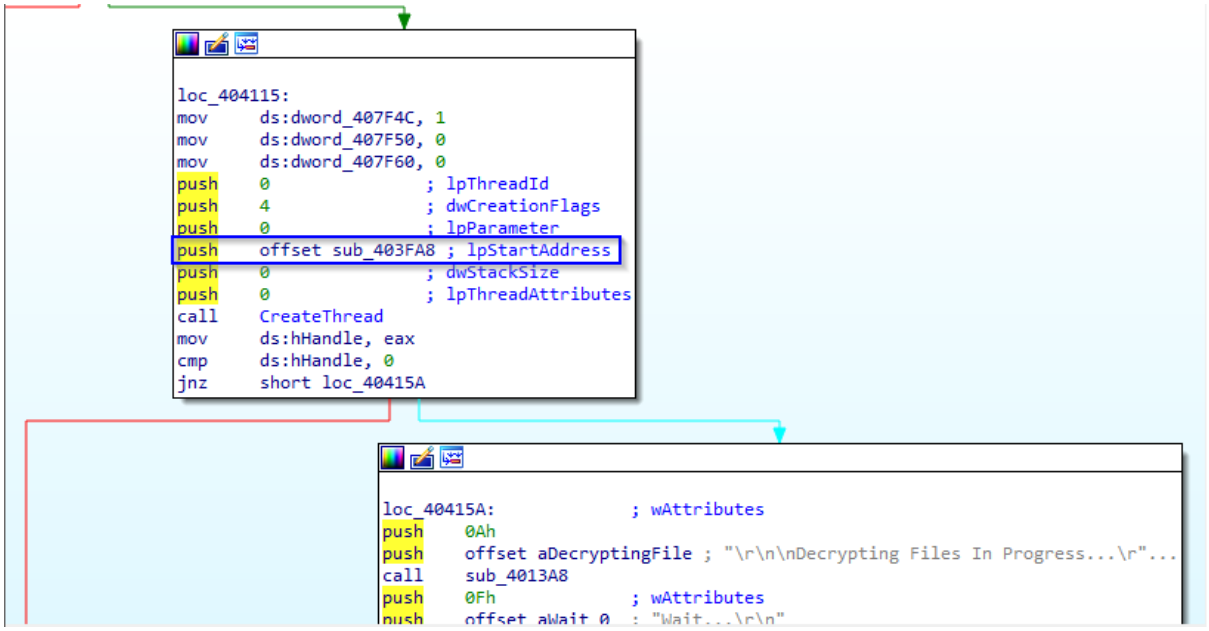
Address	Hex																ASCII
007AEB5C	BD	74	00	1C	F8	EB	7A	00	50	E7	7A	00	18	EB	7A	00	½t..øëz.Pçz..ëz.
007AEB6C	00	00	00	00	00	00	00	00	00	00	00	00	12	00	00	00	....
007AEB7C	17	00	00	00	09	00	00	00	AB	AB	AB	AB	AB	AB	AB	AB	........««««««««
007AEB8C	00	00	00	00	00	00	00	00	00	00	00	00	5D	4B	53	79	............]KSy
007AEB9C	BC	74	00	18	64	00	69	00	73	00	61	00	62	00	6C	00	¼t..d.i.s.a.b.l.
007AEBAC	65	00	77	00	69	00	6E	00	33	00	32	00	6B	00	73	00	e.w.i.n.3.2.k.s.
007AEBBC	79	00	73	00	74	00	65	00	6D	00	63	00	61	00	6C	00	y.s.t.e.m.c.a.l.
007AEBCC	6C	00	73	00	00	00	00	00	00	00	00	00	00	00	00	00	l.s.............
007AEBDC	00	00	00	00	AB	AB	AB	AB	AB	AB	AB	AB	00	00	00	00	....«««««««....
007AEBEC	00	00	00	00	5E	4B	53	7A	BF	74	00	1C	80	EC	7A	00	....^KSz¿t...ìz.
007AEBFC	60	EB	7A	00	A0	EB	7A	00	00	00	00	00	00	00	00	00	`ëz. ëz.........
007AEC0C	00	00	00	00	18	00	00	00	1F	00	00	00	0A	00	00	00	....

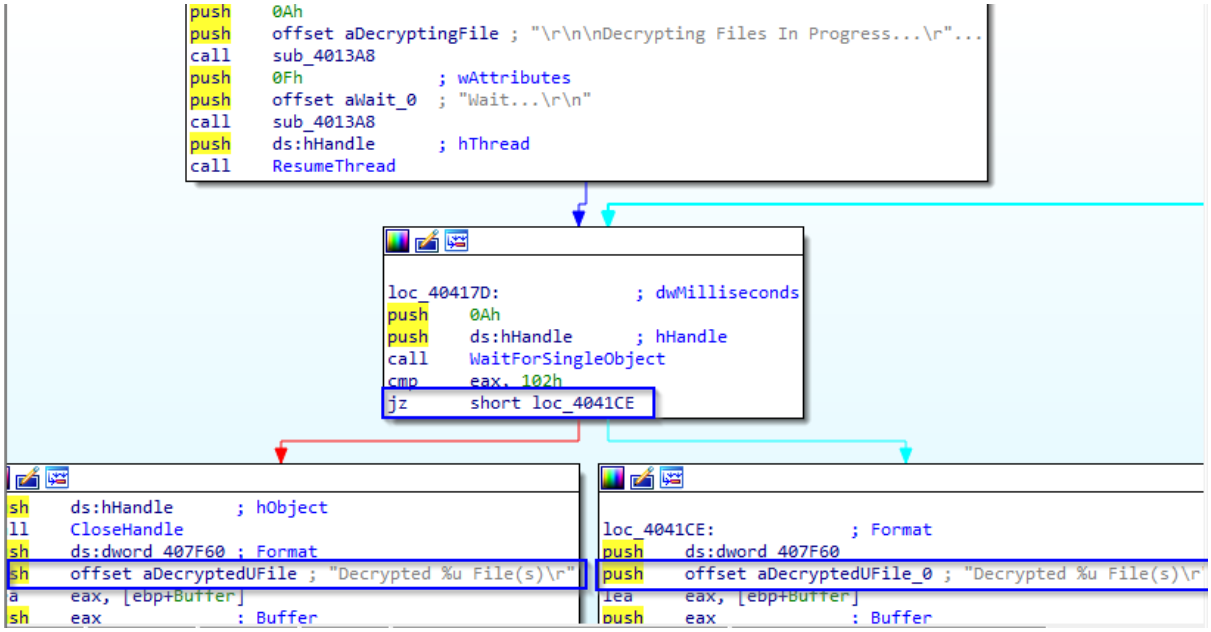
In the context of the analysis of the file %s.README.txt (where %s stands for string variable which is related to a pattern contained the filename) by the decryptor, is called the function RtlFreeHeap to free a memory block allocated from an heap.

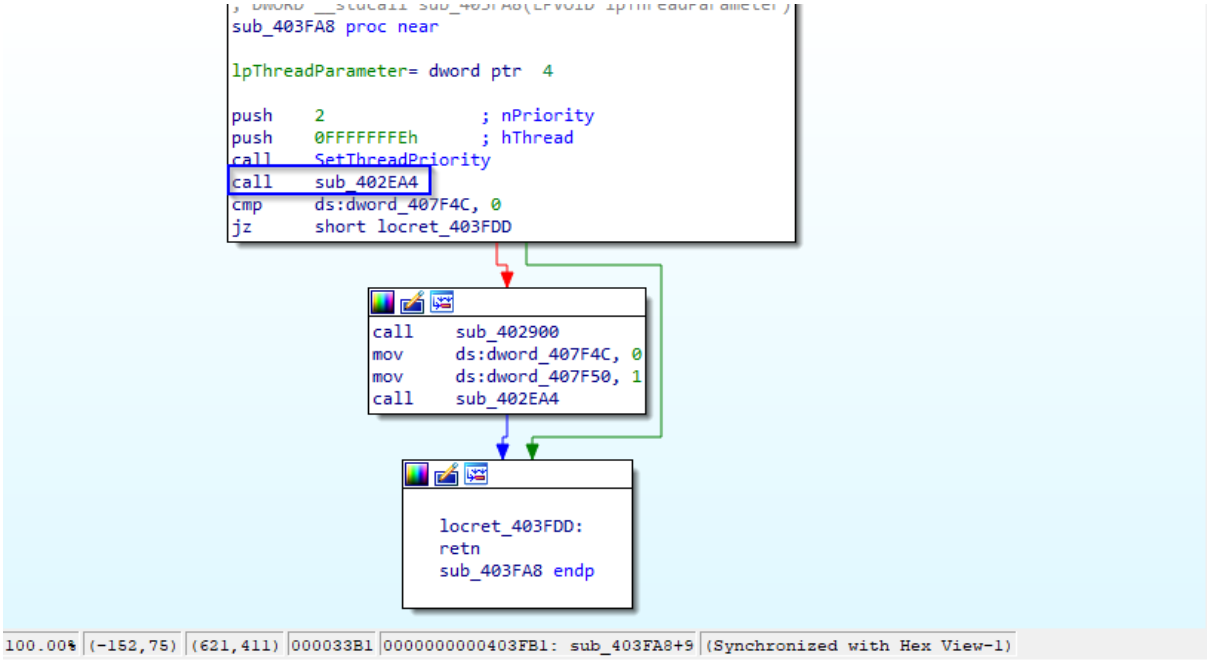


It is important to underline that, in the context of files loop gathering to obtain the details of the files to decrypt on the disks of the compromised machine, it is created an execution thread

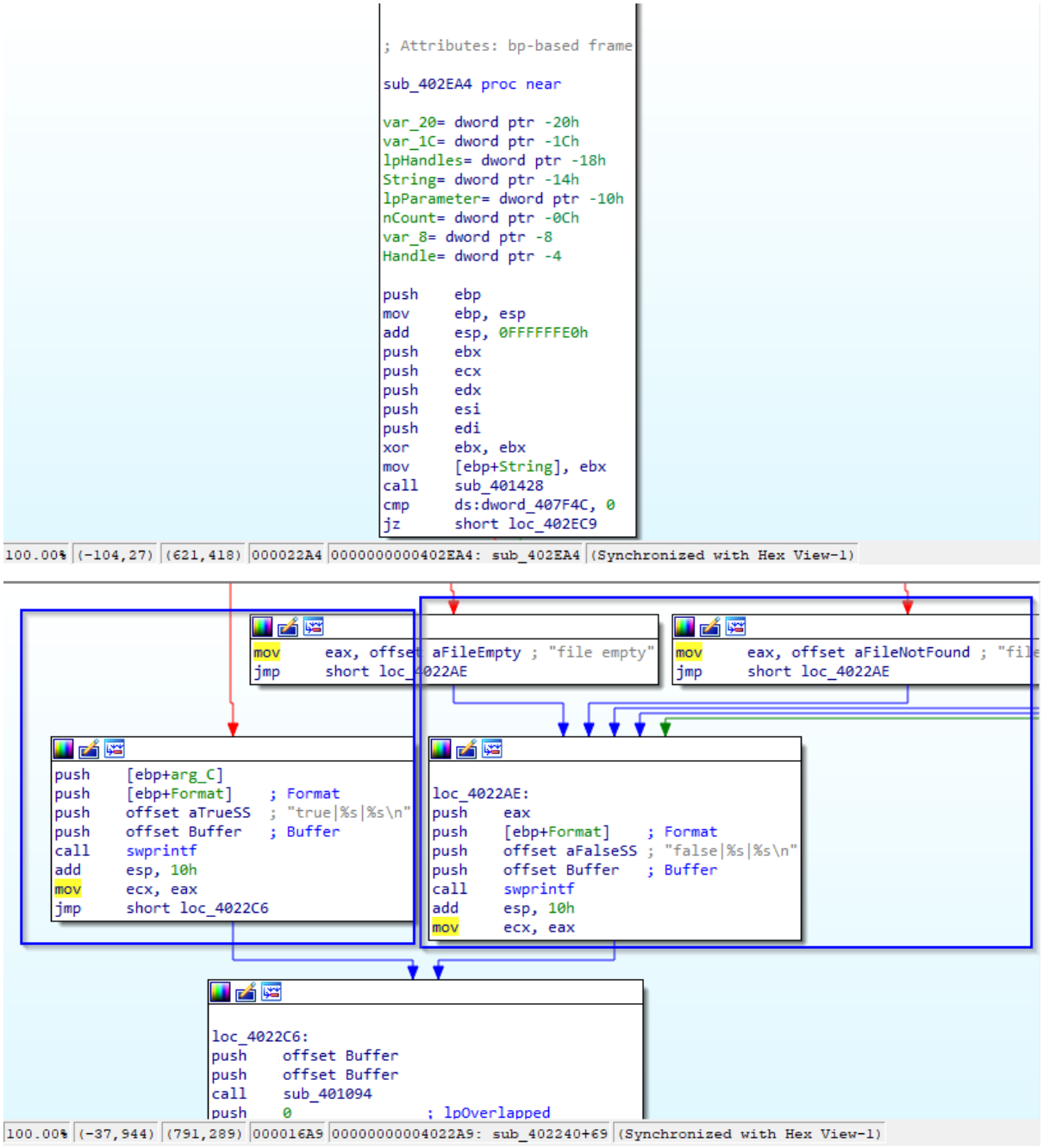
which points to the function **sub_403FA8**, which calls the function **sub_402EA4**. After the execution of the functions of the thread is present the instruction **jz short loc_4041CE** and the "switch" is managed through the offsets **aDecryptedUFile** and **aDecryptedUFile_0** based on the result get from the thread execution:

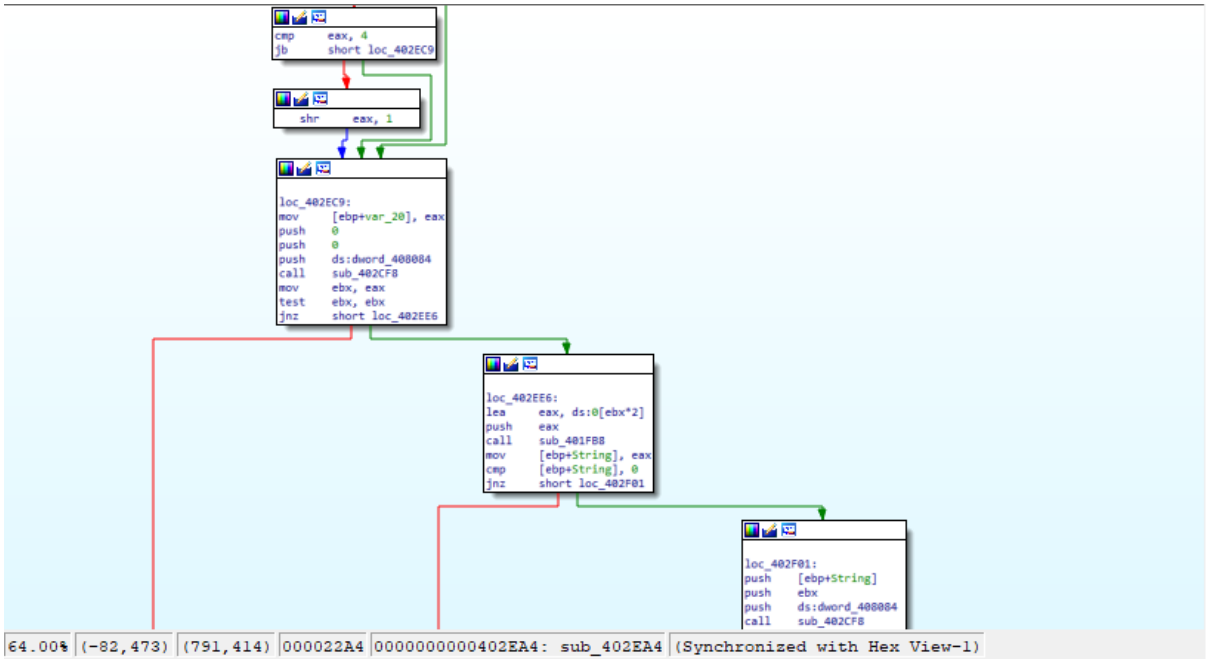


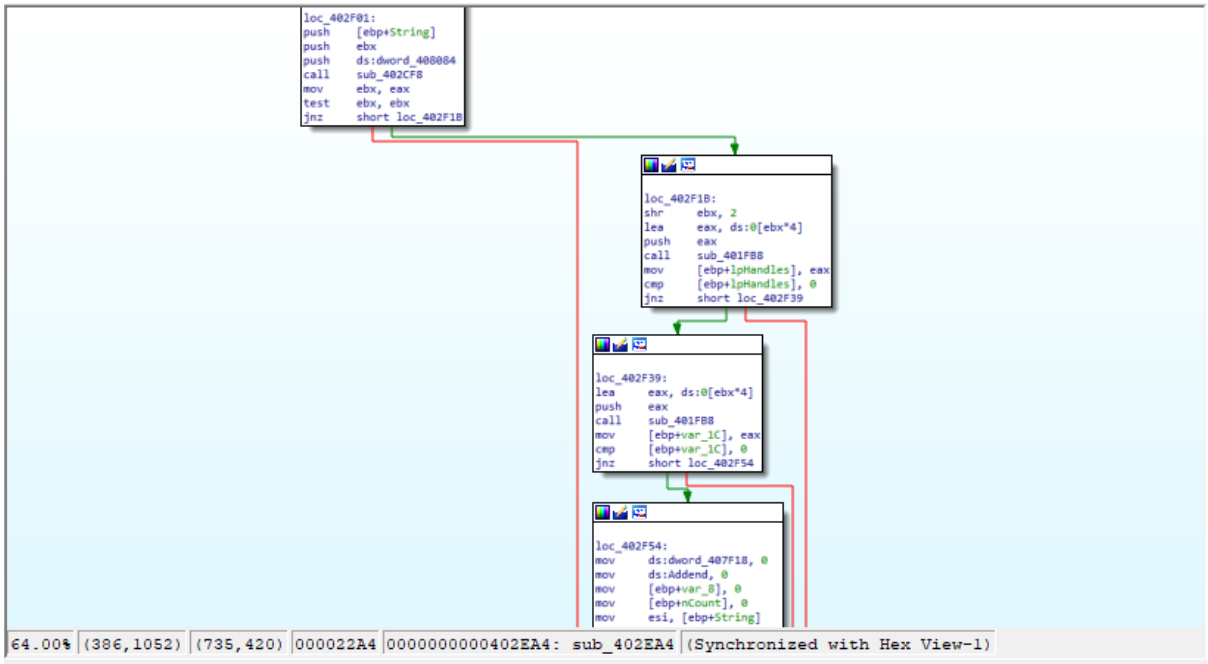




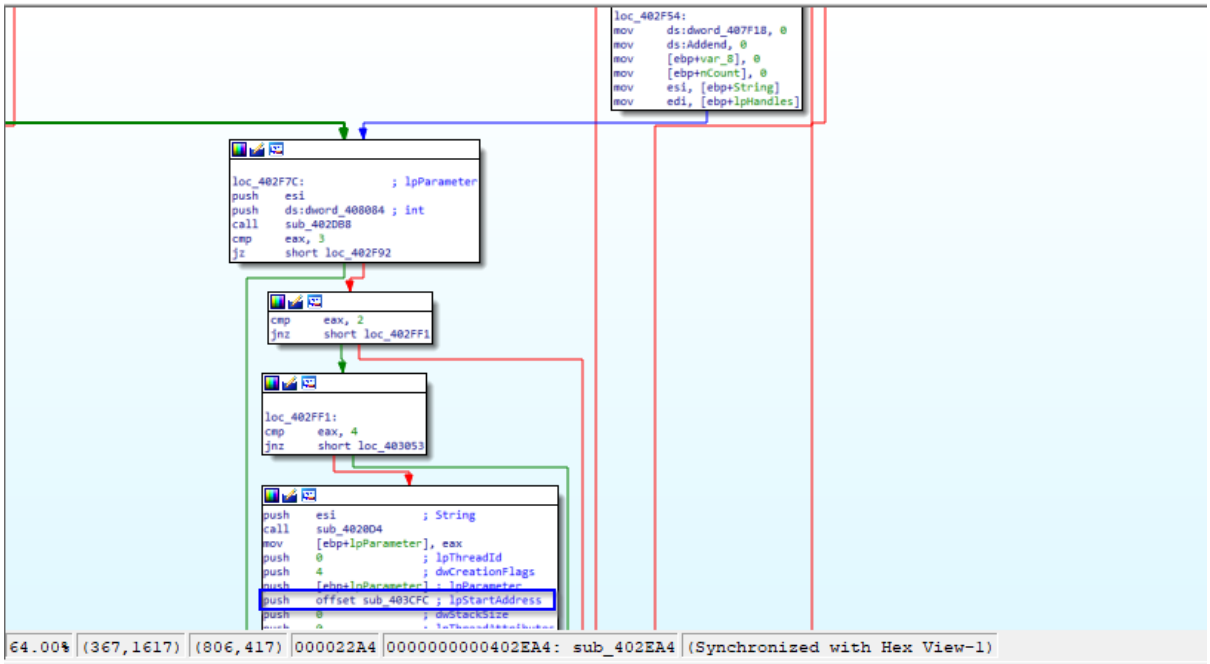
The function **sub_402EA4** is fundamental because a check operation is performed with a sort of "execution key", infact it is present a chain of cmp and jnz instructions that point to others labels (for example loc_402F01).

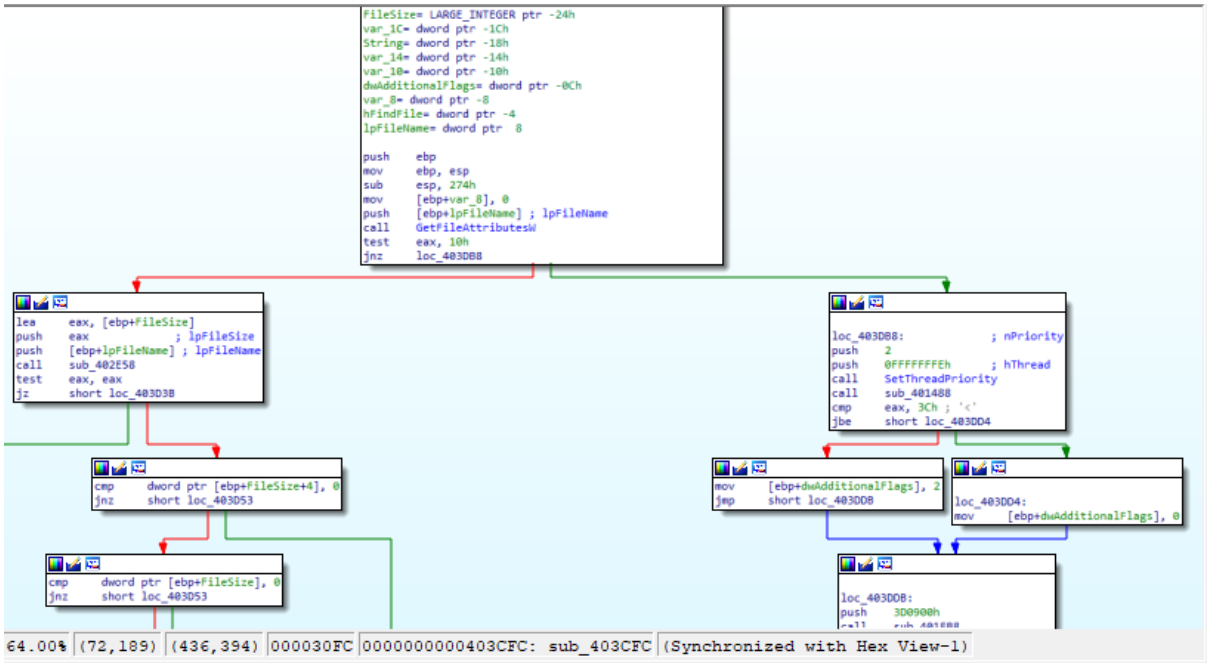

; Attributes: bp-based frame
sub_402EA4 proc near
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
lpHandles= dword ptr -18h
String= dword ptr -14h
lpParameter= dword ptr -10h
nCount= dword ptr -0Ch
var_8= dword ptr -8
Handle= dword ptr -4
push ebp
mov ebp, esp
add esp, 0FFFFFFE0h
push ebx
push ecx
push edx
push esi
push edi
xor ebx, ebx
mov [ebp+String], ebx
call sub_401428
cmp ds:dword_407F4C, 0
jz short loc_402EC9
100.00% (-104,27) (621,418) 000022A4 00000000000402EA4: sub_402EA4 (Synchronized with Hex View-1)
mov eax, offset aFileEmpty ; "file empty"
jmp short loc_4022AE
mov eax, offset aFileNotFound ; "file
jmp short loc_4022AE
push [ebp+arg_C]
push [ebp+Format] ; Format
push offset aTrueSS ; "true|%s|%s\n"
push offset Buffer ; Buffer
call swprintf
add esp, 10h
mov ecx, eax
jmp short loc_4022C6
loc_4022AE:
push eax
push [ebp+Format] ; Format
push offset aFalseSS ; "false|%s|%s\n"
push offset Buffer ; Buffer
call swprintf
add esp, 10h
mov ecx, eax
loc_4022C6:
push offset Buffer
push offset Buffer
call sub_401094
push 0 ; lpOverlapped
100.00% (-37,944) (791,289) 000016A9 00000000004022A9: sub_402240+69 (Synchronized with Hex View-1)

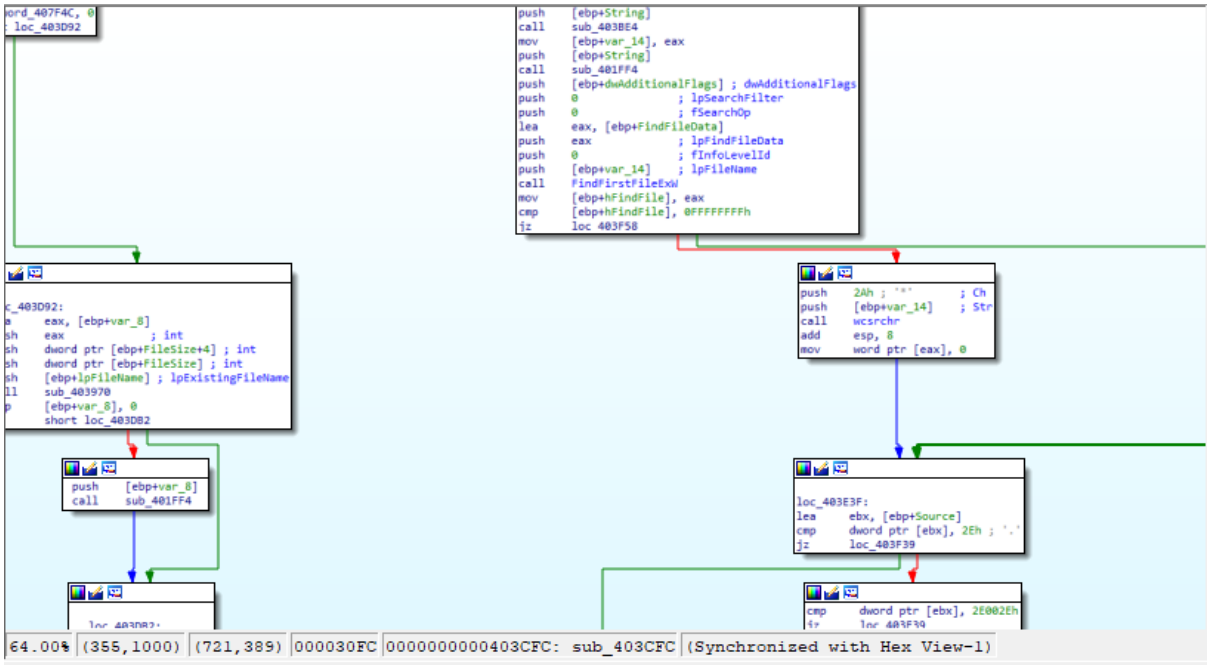


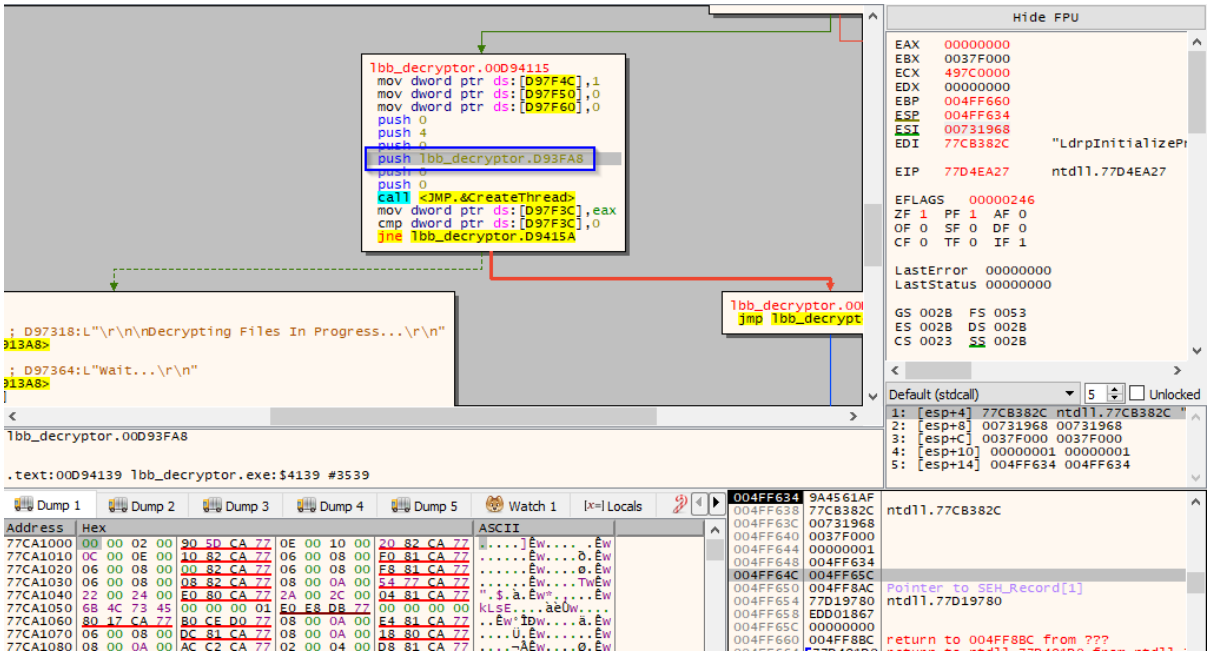


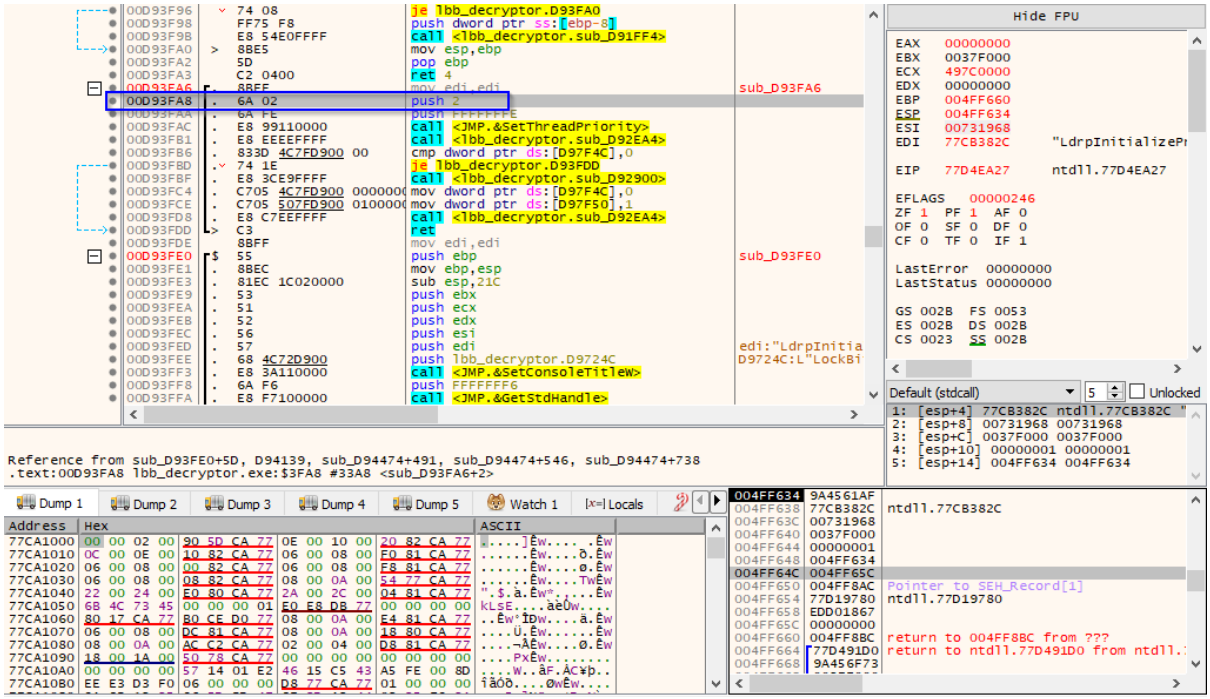
After the execution of the check instructions and validation we arrive to the point of the execution which calls the function **sub_403CFC**, which effectively starts the operations of files gathering (note the function **FindFirstFileExW** and the wildcard "*"), and then continue with the obtaining of the attributes of the encrypted files:

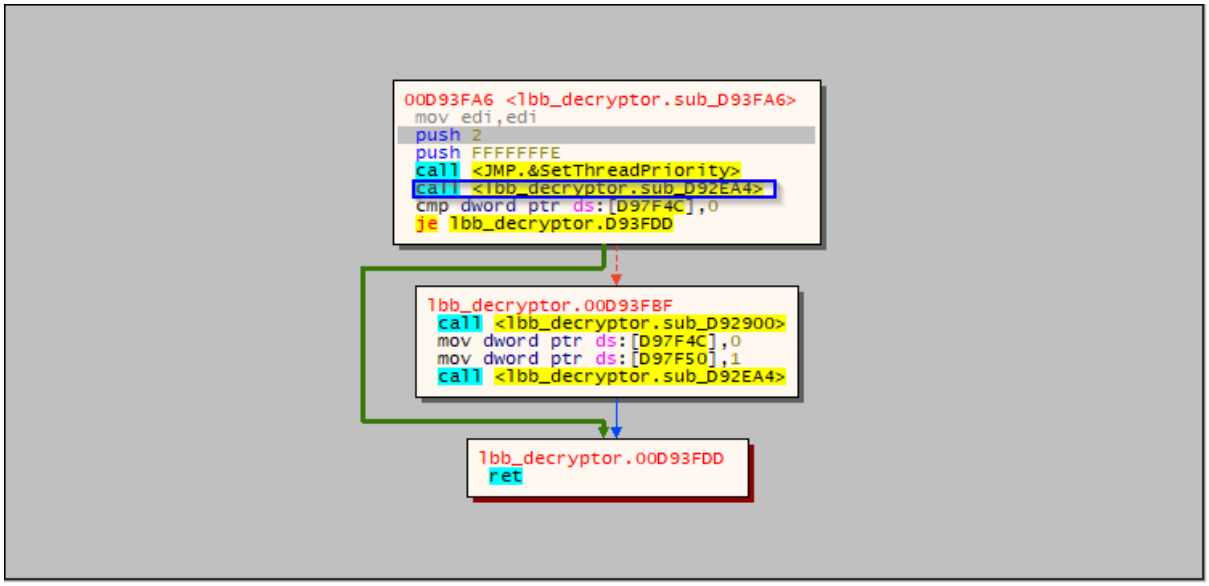

loc_402F54:
mov ds:dword_407F18, 0
mov ds:Addend, 0
mov [ebp+var_8], 0
mov [ebp+nCount], 0
mov esi, [ebp+String]
mov edi, [ebp+lpHandles]
loc_402F7C: ; lpParameter
push esi
push ds:dword_408084 ; int
call sub_402DB8
cmp eax, 3
jz short loc_402F92
cmp eax, 2
jnz short loc_402FF1
loc_402FF1:
cmp eax, 4
jnz short loc_403053
push esi ; String
call sub_4020D4
mov [ebp+lpParameter], eax
push 0 ; lpThreadId
push 4 ; dwCreationFlags
push [ebp+lpParameter] ; lpParameter
push offset sub_403CFC ; lpStartAddress
push 0 ; dwStackSize
64.00% (367,1617) (806,417) 000022A4 0000000000402EA4: sub_402EA4 (Synchronized with Hex View-1)

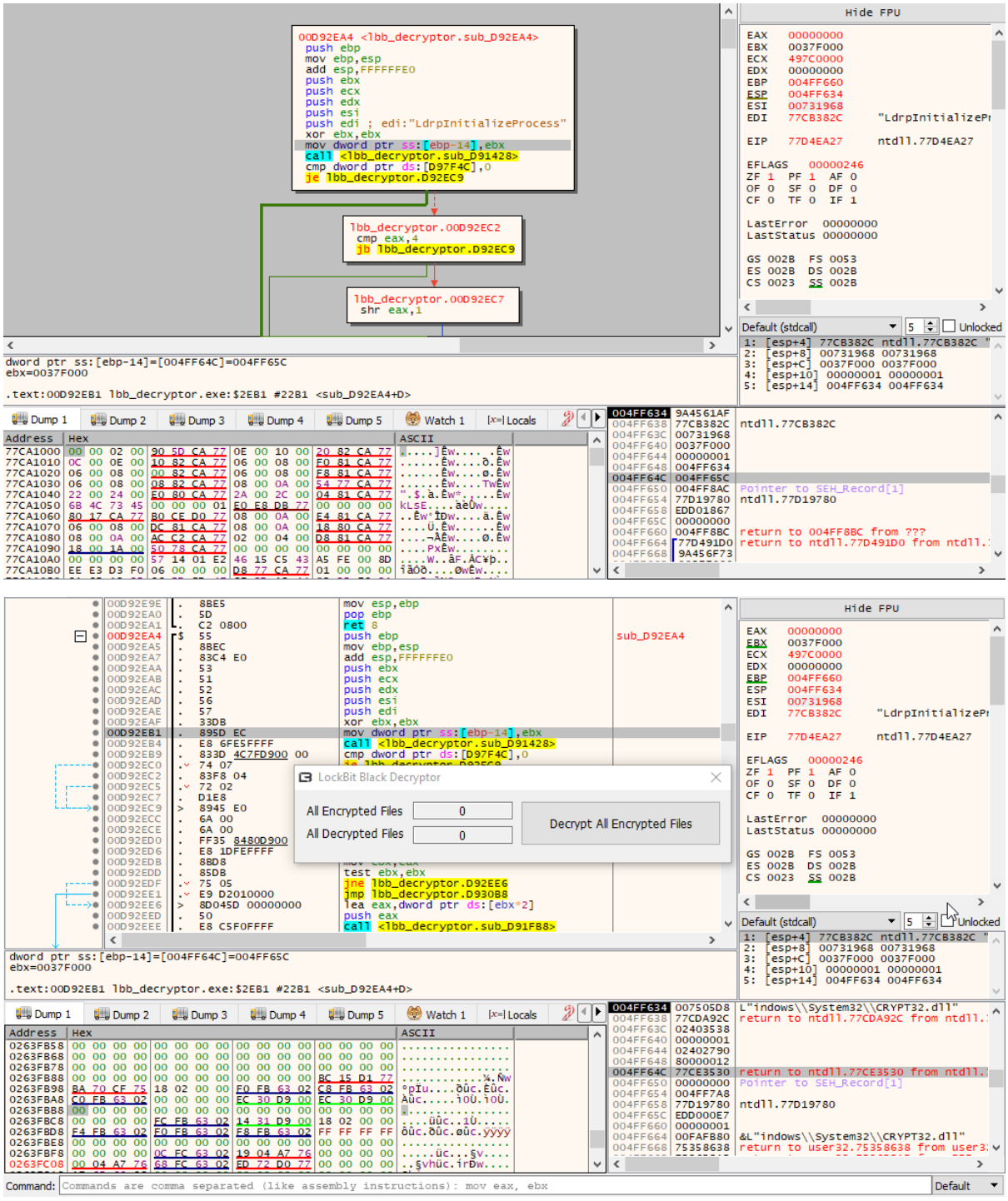

FileSize= LARGE_INTEGER ptr -24h
var_1C= dword ptr -1Ch
String= dword ptr -18h
var_14= dword ptr -14h
var_10= dword ptr -10h
dwAdditionalFlags= dword ptr -0Ch
var_8= dword ptr -8
hFindFile= dword ptr -4
lpFileName= dword ptr 8
push ebp
mov ebp, esp
sub esp, 274h
mov [ebp+var_8], 0
push [ebp+lpFileName] ; lpFileName
call GetFileAttributesW
test eax, 10h
jnz loc_403DB8
lea eax, [ebp+FileSize]
push eax ; lpFileSize
push [ebp+lpFileName] ; lpFileName
call sub_402E58
test eax, eax
jz short loc_403D3B
cmp dword ptr [ebp+FileSize+4], 0
jnz short loc_403D53
cmp dword ptr [ebp+FileSize], 0
jnz short loc_403D53
loc_403DB8: ; nPriority
push 2
push 0FFFFFFFEh ; hThread
call SetThreadPriority
call sub_401488
cmp eax, 3Ch ; '<'
jbe short loc_403DD4
mov [ebp+dwAdditionalFlags], 2
jmp short loc_403DDB
loc_403DD4:
mov [ebp+dwAdditionalFlags], 0
loc_403DDB:
64.00% (72,189) (436,394) 000030FC 0000000000403CFC: sub_403CFC (Synchronized with Hex View-1)

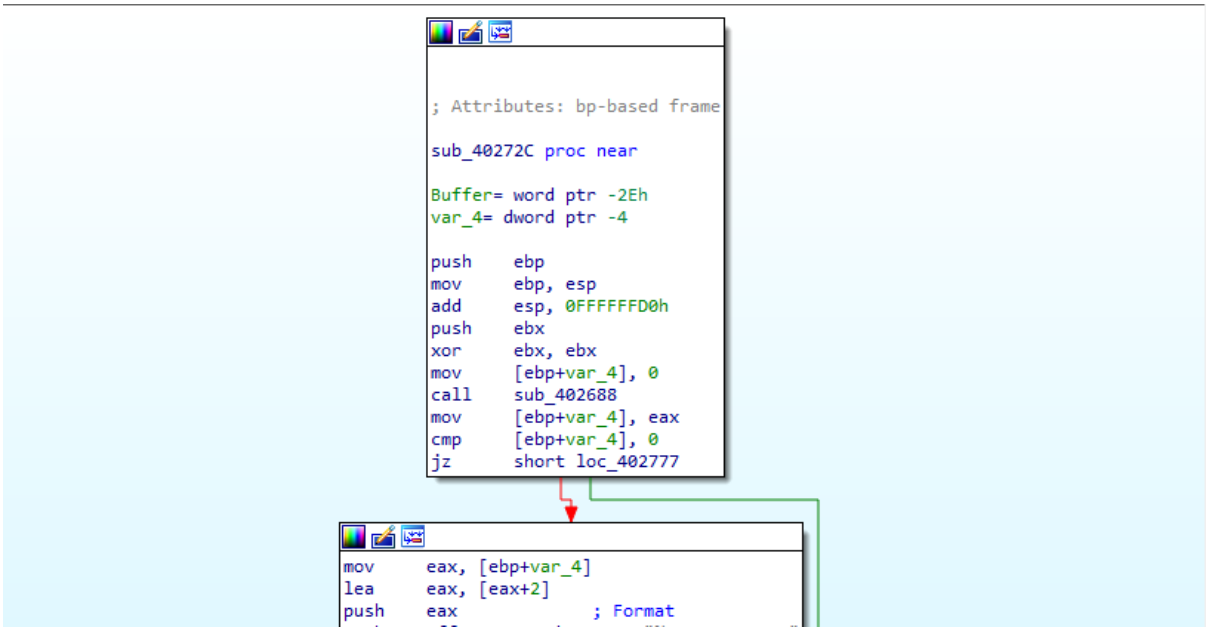

push [ebp+String]
call sub_403BE4
mov [ebp+var_14], eax
push [ebp+String]
call sub_401FF4
push [ebp+dwAdditionalFlags] ; dwAdditionalFlags
push 0 ; lpSearchFilter
push 0 ; fSearchOp
lea eax, [ebp+FindFileData]
push eax ; lpFindFileData
push 0 ; fInfoLevelId
push [ebp+var_14] ; lpFileName
call FindFirstFileExW
mov [ebp+hFindFile], eax
cmp [ebp+hFindFile], 0FFFFFFFFh
jz loc_403F58
push [ebp+var_8]
call sub_401FF4
push 2Ah ; '*'
push [ebp+var_14]
call wcsrchr
add esp, 8
mov word ptr [eax], 0
loc_403E3F:
lea ebx, [ebp+Source]
cmp dword ptr [ebx], 2Eh ; '.'
jz loc_403F39
64.00% (355,1000) (721,389) 000030FC 0000000000403CFC: sub_403CFC (Synchronized with Hex View-1)


lbb_decryptor.00D94115
mov dword ptr ds:[D97F4C],1
mov dword ptr ds:[D97F50],0
mov dword ptr ds:[D97F60],0
push 0
push 4
push lbb_decryptor.D93FA8
push 0
call <JMP.&CreateThread>
mov dword ptr ds:[D97F3C],eax
cmp dword ptr ds:[D97F3C],0
jne lbb_decryptor.D9415A
; D97318:L"\r\n\nDecrypting Files In Progress...\r\n"
; D97364:L"Wait...\r\n"
lbb_decryptor.00D93FA8
.text:00D94139 lbb_decryptor.exe:$4139 #3539
Hide FPU
EAX 00000000
EBX 0037F000
ECX 497C0000
EDX 00000000
EBP 004FF660
ESP 004FF634
ESI 00731968
EDI 77CB382C "LdrpInitializeP
EIP 77D4EA27 ntdll.77D4EA27
EFLAGS 00000246
ZF 1 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1
LastError 00000000
LastStatus 00000000
GS 002B FS 0053
ES 002B DS 002B
CS 0023 SS 002B
Default (stdcall)
1: [esp+4] 77CB382C ntdll.77CB382C
2: [esp+8] 00731968 00731968
3: [esp+C] 0037F000 0037F000
4: [esp+10] 00000001 00000001
5: [esp+14] 004FF634 004FF634
Dump 1
Dump 2
Dump 3
Dump 4
Dump 5
Watch 1
Locals
Address Hex ASCII
77CA1000 00 00 02 00 90 5D CA 77 0E 00 10 00 20 82 CA 77
77CA1010 0C 00 0E 00 10 82 CA 77 06 00 08 00 F0 81 CA 77
77CA1020 06 00 08 00 00 82 CA 77 06 00 08 00 F8 81 CA 77
77CA1030 06 00 08 00 08 82 CA 77 08 00 0A 00 54 77 CA 77
77CA1040 22 00 24 00 E0 80 CA 77 2A 00 2C 00 04 81 CA 77
77CA1050 6B 4C 73 45 00 00 00 01 E0 E8 DB 77 00 00 00 00
77CA1060 80 17 CA 77 B0 CE D0 77 08 00 0A 00 E4 81 CA 77
77CA1070 06 00 08 00 DC 81 CA 77 08 00 0A 00 18 80 CA 77
77CA1080 08 00 0A 00 AC C2 CA 77 02 00 04 00 D8 81 CA 77
004FF634 9A4561AF
004FF638 77CB382C ntdll.77CB382C
004FF63C 00731968
004FF640 0037F000
004FF644 00000001
004FF648 004FF634
004FF64C 004FF65C
004FF650 004FF8AC Pointer to SEH_Record[1]
004FF654 77D19780 ntdll.77D19780
004FF658 EDD01867
004FF65C 00000000
004FF660 004FF8BC return to 004FF8BC from ???

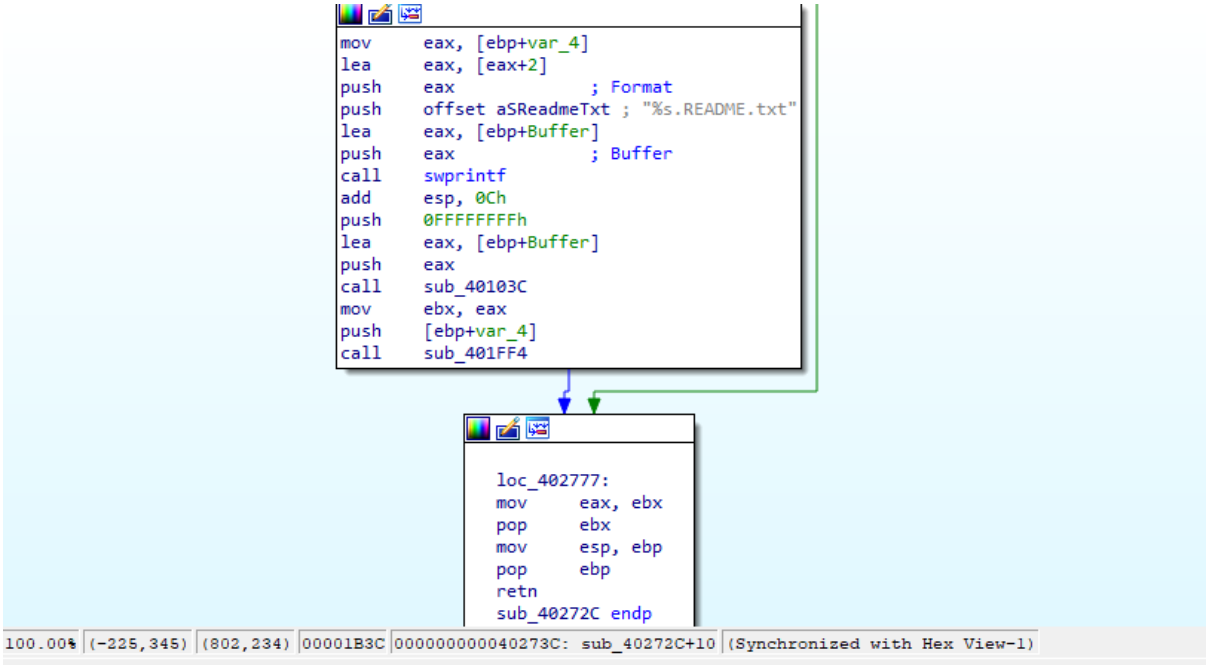


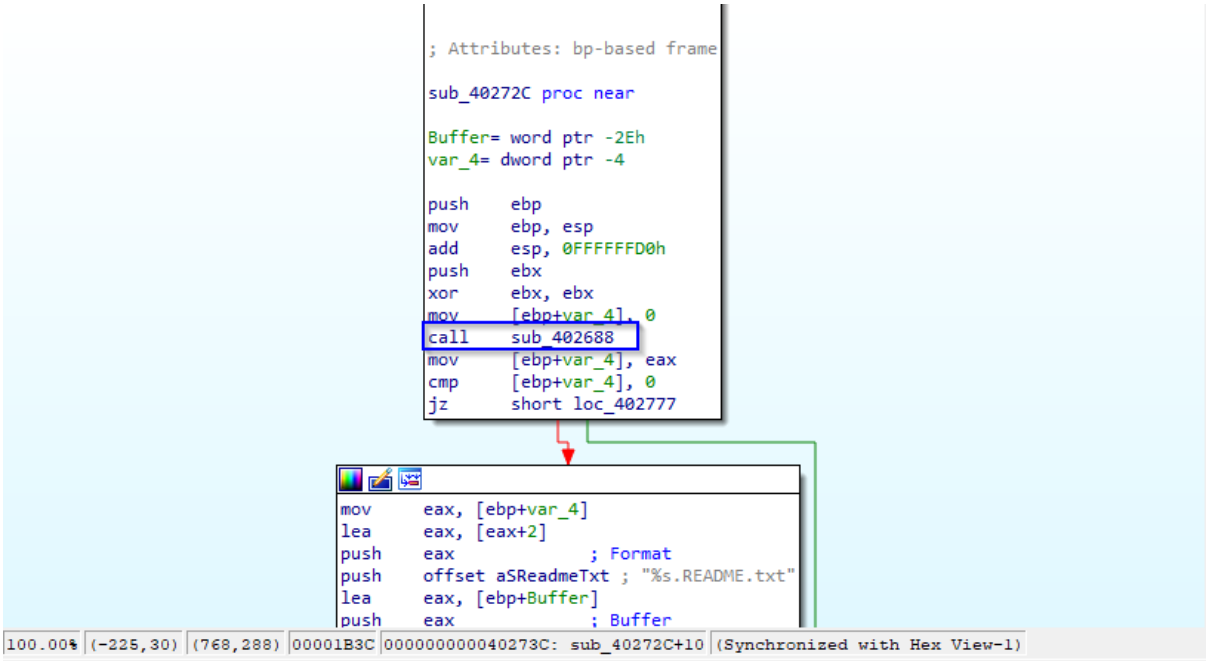

00D93FA6 <lbb_decryptor.sub_D93FA6>
mov edi,edi
push 2
push FFFFFFFE
call <JMP.&SetThreadPriority>
call <lbb_decryptor.sub_D92EA4>
cmp dword ptr ds:[D97F4C],0
je lbb_decryptor.D93FDD
lbb_decryptor.00D93FBF
call <lbb_decryptor.sub_D92900>
mov dword ptr ds:[D97F4C],0
mov dword ptr ds:[D97F50],1
call <lbb_decryptor.sub_D92EA4>
lbb_decryptor.00D93FDD
ret

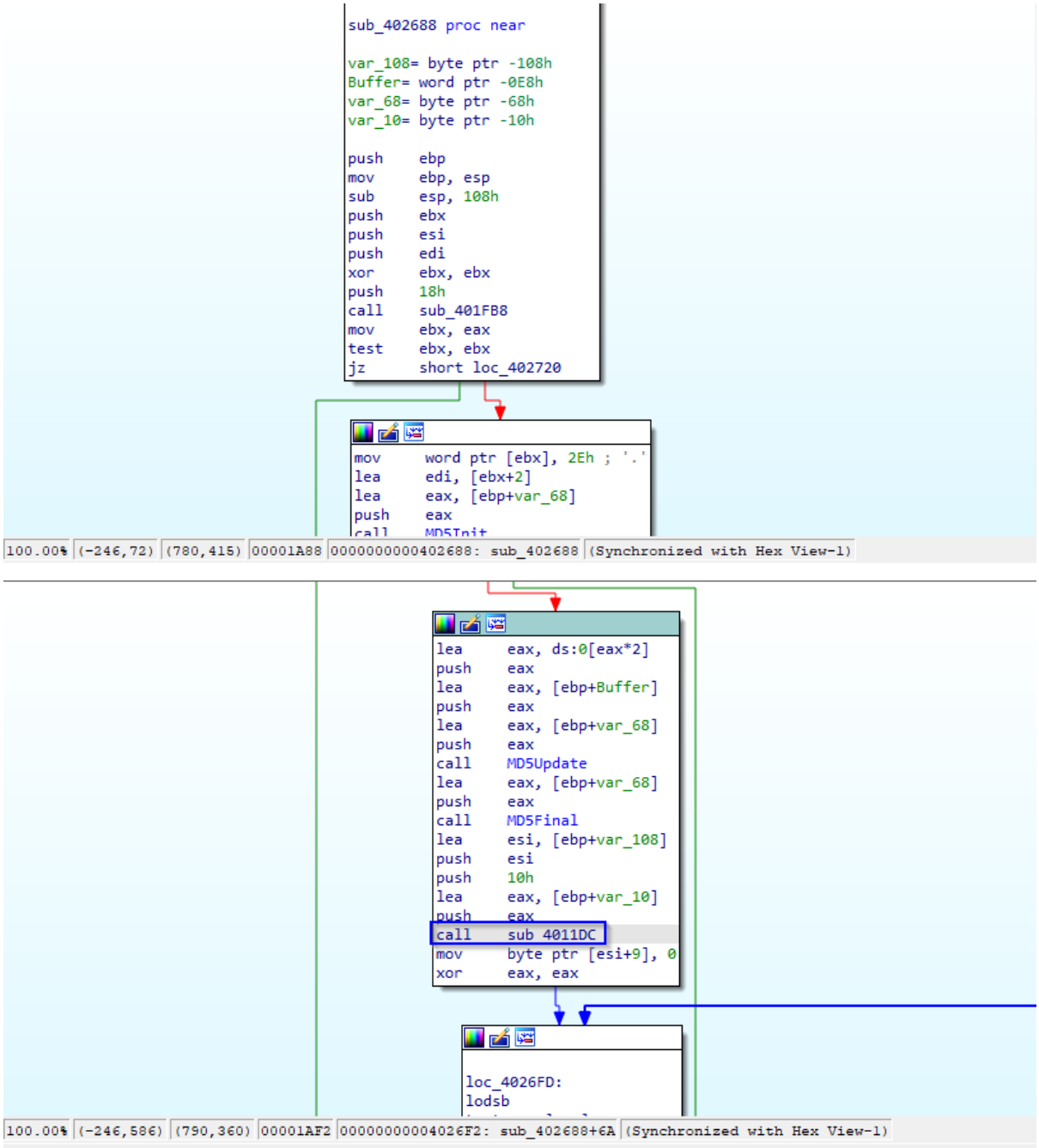

00D92EA4 <lbb_decryptor.sub_D92EA4>
push ebp
mov ebp,esp
add esp,FFFFFFE0
push ebx
push ecx
push edx
push esi
push edi ; edi:"LdrpInitializeProcess"
xor ebx,ebx
mov dword ptr ss:[ebp-14],ebx
call <lbb_decryptor.sub_D91428>
cmp dword ptr ds:[D97F4C],0
je lbb_decryptor.D92EC9
lbb_decryptor.00D92EC2
cmp eax,4
jb lbb_decryptor.D92EC9
lbb_decryptor.00D92EC7
shr eax,1
Hide FPU
EAX 00000000
EBX 0037F000
ECX 497C0000
EDX 00000000
EBP 004FF660
ESP 004FF634
ESI 00731968
EDI 77CB382C "LdrpInitializePr
EIP 77D4EA27 ntdll.77D4EA27
EFLAGS 00000246
ZF 1 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1
LastError 00000000
LastStatus 00000000
GS 002B FS 0053
ES 002B DS 002B
CS 0023 SS 002B
Default (stdcall)
Unlocked
1: [esp+4] 77CB382C ntdll.77CB382C
2: [esp+8] 00731968 00731968
3: [esp+C] 0037F000 0037F000
4: [esp+10] 00000001 00000001
5: [esp+14] 004FF634 004FF634
dword ptr ss:[ebp-14]=[004FF64C]=004FF65C
ebx=0037F000
.text:00D92EB1 lbb_decryptor.exe:$2EB1 #22B1 <sub_D92EA4+D>
Dump 1
Dump 2
Dump 3
Dump 4
Dump 5
Watch 1
Locals
Address
Hex
ASCII
004FF634 9A4561AF
004FF638 77CB382C ntdll.77CB382C
004FF63C 00731968
004FF640 0037F000
004FF644 00000001
004FF648 004FF634
004FF64C 004FF65C
004FF650 004FF8AC Pointer to SEH_Record[1]
004FF654 77D19780 ntdll.77D19780
004FF658 EDD01867
004FF65C 00000000
004FF660 004FF8BC
004FF664 77D491D0 return to ntdll.77D491D0 from ntdll.
004FF668 9A456F73
00D92E9E . 8BE5 mov esp,ebp
00D92EA0 . 5D pop ebp
00D92EA1 . C2 0800 ret 8
00D92EA4 r$ 55 push ebp sub_D92EA4
00D92EA5 . 8BEC mov ebp,esp
00D92EA7 . 83C4 E0 add esp,FFFFFFE0
00D92EAA . 53 push ebx
00D92EAB . 51 push ecx
00D92EAC . 52 push edx
00D92EAD . 56 push esi
00D92EAE . 57 push edi
00D92EAF . 33DB xor ebx,ebx
00D92EB1 . 895D EC mov dword ptr ss:[ebp-14],ebx
00D92EB4 . E8 6FE5FFFF call <lbb_decryptor.sub_D91428>
00D92EB9 . 833D 4C7FD900 00 cmp dword ptr ds:[D97F4C],0
00D92EC0 .v 74 07
00D92EC2 . 83F8 04
00D92EC5 .v 72 02
00D92EC7 . D1E8
00D92EC9 > 8945 E0
00D92ECC . 6A 00
00D92ECE . 6A 00
00D92ED0 . FF35 8480D900
00D92ED6 . E8 1DFEFFFF
00D92EDB . 8BD8
00D92EDD . 85DB test ebx,ebx
00D92EDF .v 75 05 jne lbb_decryptor.D92EE6
00D92EE1 .v E9 D2010000 jmp lbb_decryptor.D93088
00D92EE6 > 8D045D 00000000 lea eax,dword ptr ds:[ebx*2]
00D92EED . 50 push eax
00D92EEE . E8 C5F0FFFF call <lbb_decryptor.sub_D91FB8>
LockBit Black Decryptor
All Encrypted Files 0
All Decrypted Files 0
Decrypt All Encrypted Files
0263FB58 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0263FB68 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0263FB78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0263FB88 00 00 00 00 00 00 00 00 BC 15 D1 77
0263FB98 BA 70 CF 75 18 02 00 00 F0 FB 63 02 C8 FB 63 02
0263FBA8 C0 FB 63 02 00 00 00 00 EC 30 D9 00 EC 30 D9 00
0263FBB8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0263FBC8 00 00 00 00 FC FB 63 02 14 31 D9 00 18 02 00 00
0263FBD8 F4 FB 63 02 F0 FB 63 02 F8 FB 63 02 FF FF FF FF
0263FBE8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0263FBF8 00 00 00 00 0C FC 63 02 19 04 A7 76 00 00 00 00
0263FC08 00 04 A7 76 68 FC 63 02 ED 72 D0 77 00 00 00 00
004FF634 007505D8 L"indows\\System32\\CRYPT32.dll"
004FF638 77CDA92C return to ntdll.77CDA92C from ntdll.
004FF63C 02403538
004FF640 00000001
004FF644 02402790
004FF648 80000012
004FF64C 77CE3530 return to ntdll.77CE3530 from ntdll.
004FF650 00000000 Pointer to SEH_Record[1]
004FF654 004FF7A8
004FF658 77D19780 ntdll.77D19780
004FF65C EDD000E7
004FF660 00000001
004FF664 00FAFB80 &L"indows\\System32\\CRYPT32.dll"
004FF668 75358638 return to user32.75358638 from user3
Command: Commands are comma separated (like assembly instructions): mov eax, ebx
Default



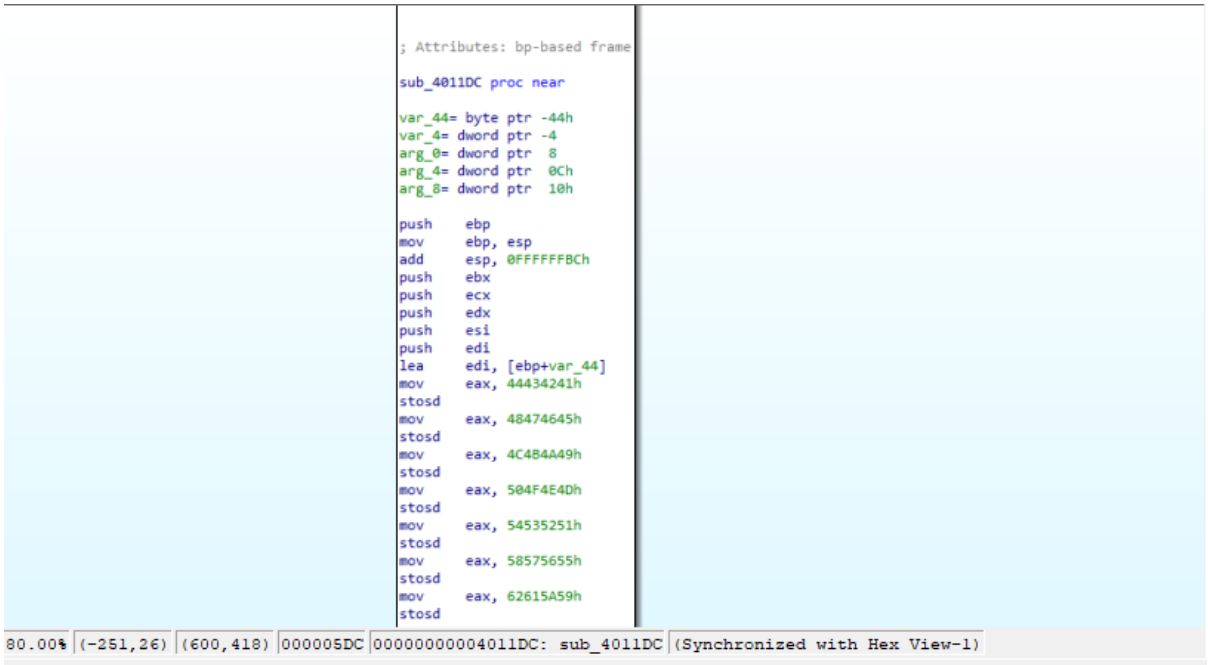
At the moment in which the content of the file README is read is possible to highlight how the function **sub_402688** is called, which consequently calls the function of hashing digest (specifically the MD5 context is created) and, then, the function **sub_4011DC** is executed:

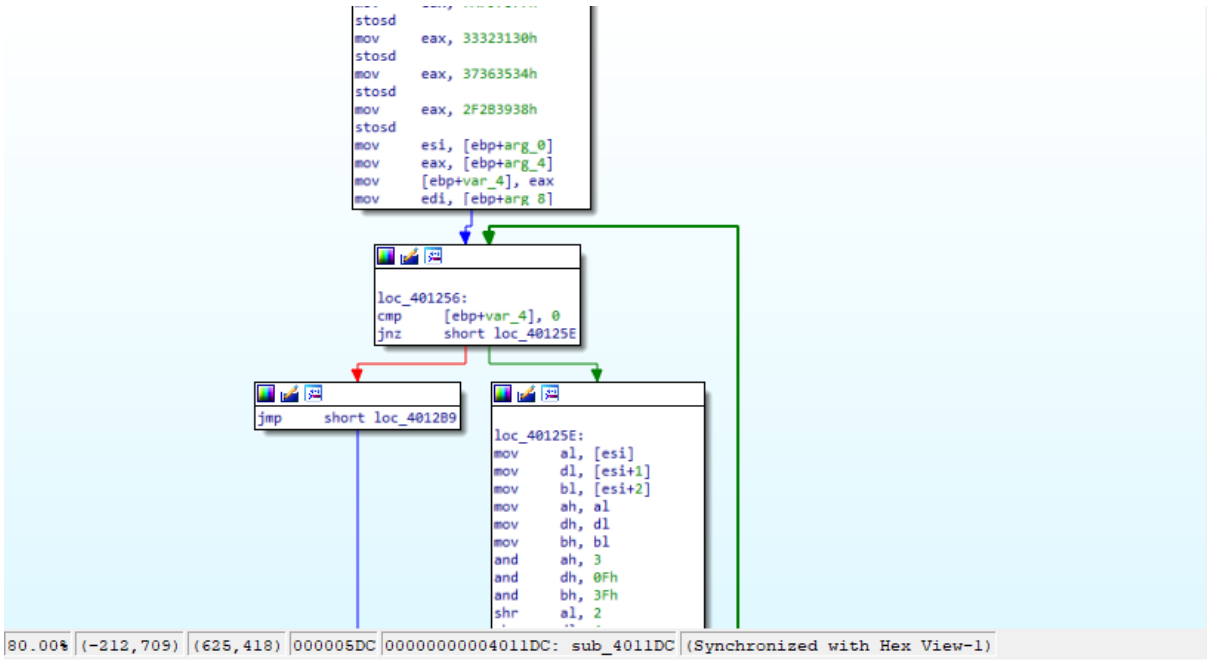

mov eax, [ebp+var_4]
lea eax, [eax+2]
push eax ; Format
push offset aSReadmeTxt ; "%s.README.txt"
lea eax, [ebp+Buffer]
push eax ; Buffer
call swprintf
add esp, 0Ch
push 0FFFFFFFFh
lea eax, [ebp+Buffer]
push eax
call sub_40103C
mov ebx, eax
push [ebp+var_4]
call sub_401FF4
loc_402777:
mov eax, ebx
pop ebx
mov esp, ebp
pop ebp
retn
sub_40272C endp
100.00% (-225,345) (802,234) 00001B3C 000000000040273C: sub_40272C+10 (Synchronized with Hex View-1)

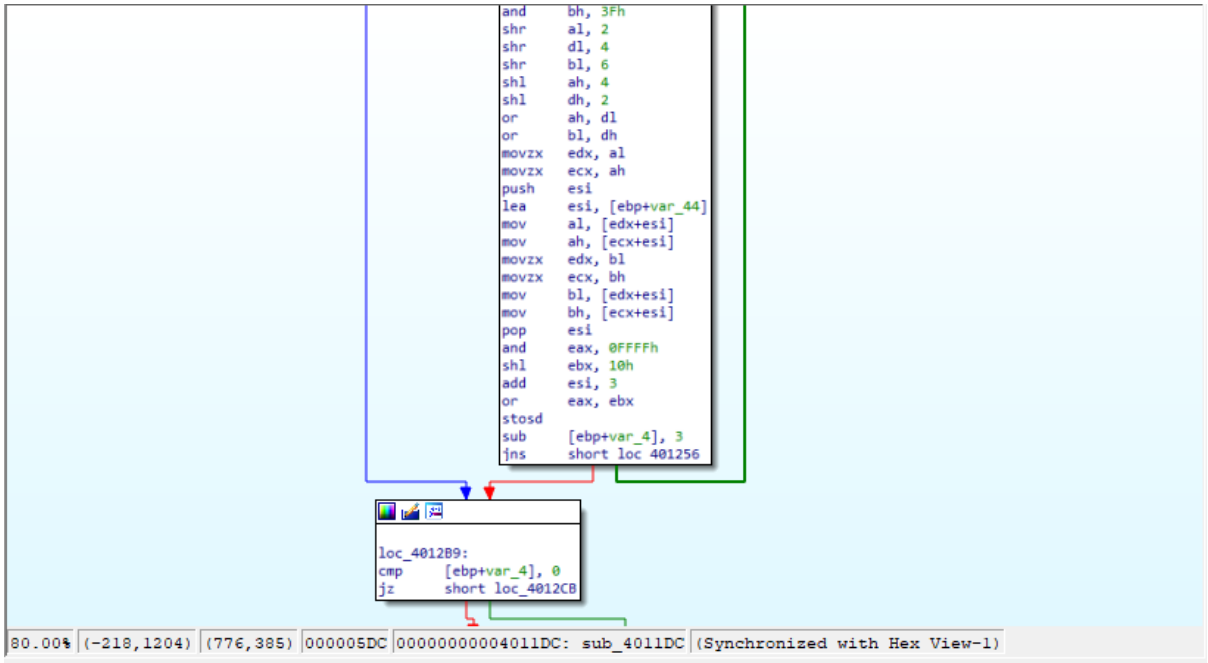

; Attributes: bp-based frame
sub_40272C proc near
Buffer= word ptr -2Eh
var_4= dword ptr -4
push ebp
mov ebp, esp
add esp, 0FFFFFFD0h
push ebx
xor ebx, ebx
mov [ebp+var_4], 0
call sub_402688
mov [ebp+var_4], eax
cmp [ebp+var_4], 0
jz short loc_402777
mov eax, [ebp+var_4]
lea eax, [eax+2]
push eax ; Format
push offset aSReadmeTxt ; "%s.README.txt"
lea eax, [ebp+Buffer]
push eax ; Buffer
100.00% (-225,30) (768,288) 00001B3C 000000000040273C: sub_40272C+10 (Synchronized with Hex View-1)

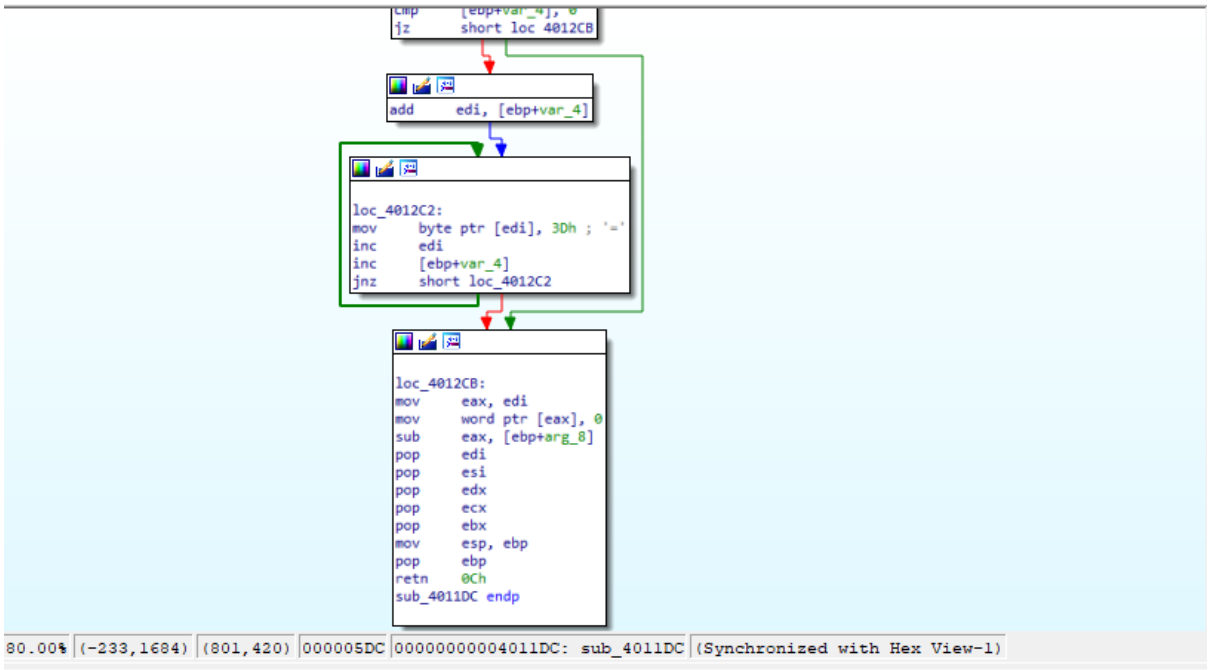


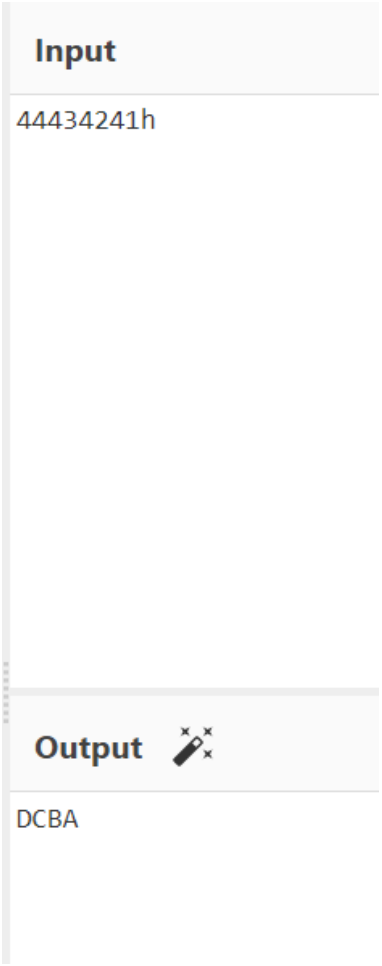
In this function there are insertions of plenty of ***hardcoded*** values which are added to the eax register. Then various AND and OR operations are executed as follows, after a compare and jnz construct to the label loc_401256. In particular the values are referred to a dictionary, in hexadecimal:


; Attributes: bp-based frame
sub_4011DC proc near
var_44= byte ptr -44h
var_4= dword ptr -4
arg_0= dword ptr 8
arg_4= dword ptr 0Ch
arg_8= dword ptr 10h
push ebp
mov ebp, esp
add esp, 0FFFFFFBCh
push ebx
push ecx
push edx
push esi
push edi
lea edi, [ebp+var_44]
mov eax, 44434241h
stosd
mov eax, 48474645h
stosd
mov eax, 4C4B4A49h
stosd
mov eax, 504F4E4Dh
stosd
mov eax, 54535251h
stosd
mov eax, 58575655h
stosd
mov eax, 62615A59h
stosd
80.00% (-251,26) (600,418) 000005DC 00000000004011DC: sub_4011DC (Synchronized with Hex View-1)

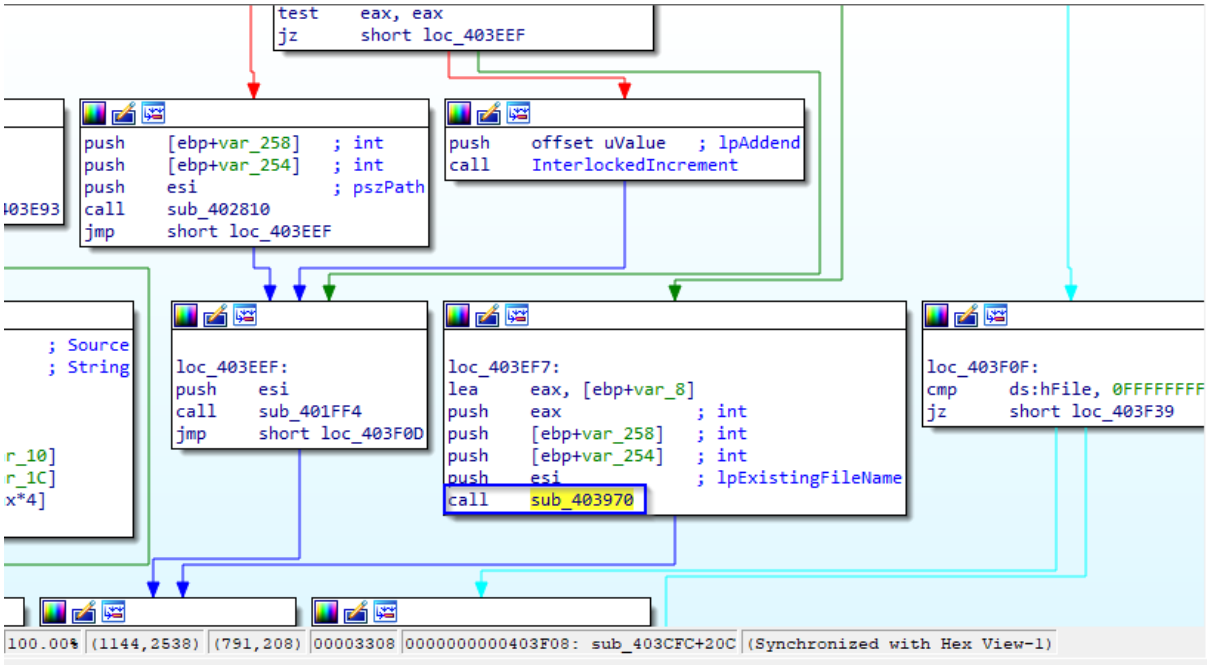

stosd
mov eax, 33323130h
stosd
mov eax, 37363534h
stosd
mov eax, 2F2B3938h
stosd
mov esi, [ebp+arg_0]
mov eax, [ebp+arg_4]
mov [ebp+var_4], eax
mov edi, [ebp+arg_8]
loc_401256:
cmp [ebp+var_4], 0
jnz short loc_40125E
jmp short loc_4012B9
loc_40125E:
mov al, [esi]
mov dl, [esi+1]
mov bl, [esi+2]
mov ah, al
mov dh, dl
mov bh, bl
and ah, 3
and dh, 0Fh
and bh, 3Fh
shr al, 2
80.00% (-212,709) (625,418) 000005DC 00000000004011DC: sub_4011DC (Synchronized with Hex View-1)

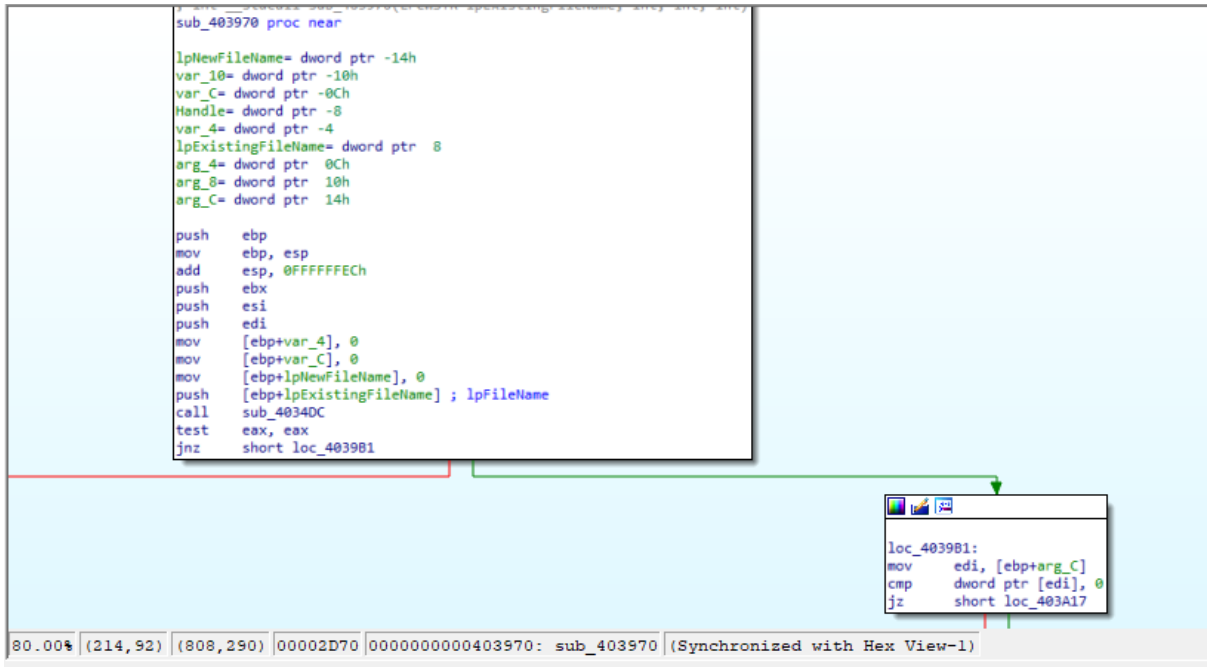

and bh, 3Fh
shr al, 2
shr dl, 4
shr bl, 6
shl ah, 4
shl dh, 2
or ah, dl
or bl, dh
movzx edx, al
movzx ecx, ah
push esi
lea esi, [ebp+var_44]
mov al, [edx+esi]
mov ah, [ecx+esi]
movzx edx, bl
movzx ecx, bh
mov bl, [edx+esi]
mov bh, [ecx+esi]
pop esi
and eax, 0FFFFh
shl ebx, 10h
add esi, 3
or eax, ebx
stosd
sub [ebp+var_4], 3
jns short loc_401256
loc_4012B9:
cmp [ebp+var_4], 0
jz short loc_4012CB
80.00% (-218,1204) (776,385) 000005DC 00000000004011DC: sub_4011DC (Synchronized with Hex View-1)

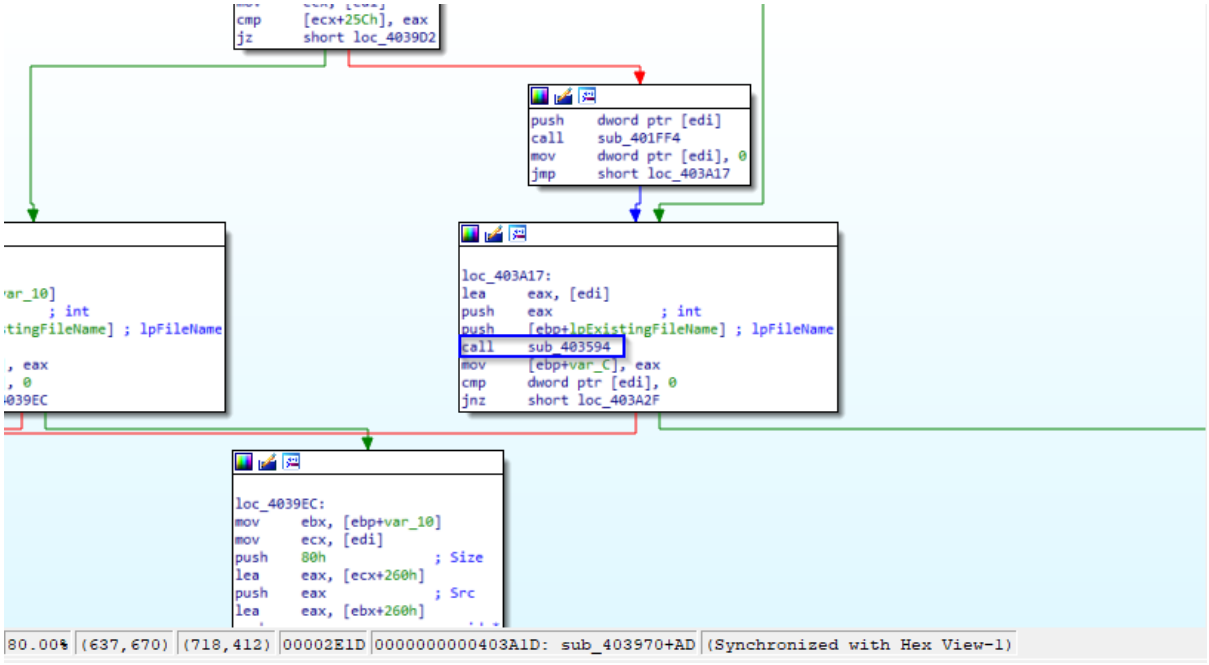

jz short loc_4012CB
add edi, [ebp+var_4]
loc_4012C2:
mov byte ptr [edi], 3Dh ; '='
inc edi
inc [ebp+var_4]
jnz short loc_4012C2
loc_4012CB:
mov eax, edi
mov word ptr [eax], 0
sub eax, [ebp+arg_8]
pop edi
pop esi
pop edx
pop ecx
pop ebx
mov esp, ebp
pop ebp
retn 0Ch
sub_4011DC endp
80.00% (-233,1684) (801,420) 000005DC 00000000004011DC: sub_4011DC (Synchronized with Hex View-1)



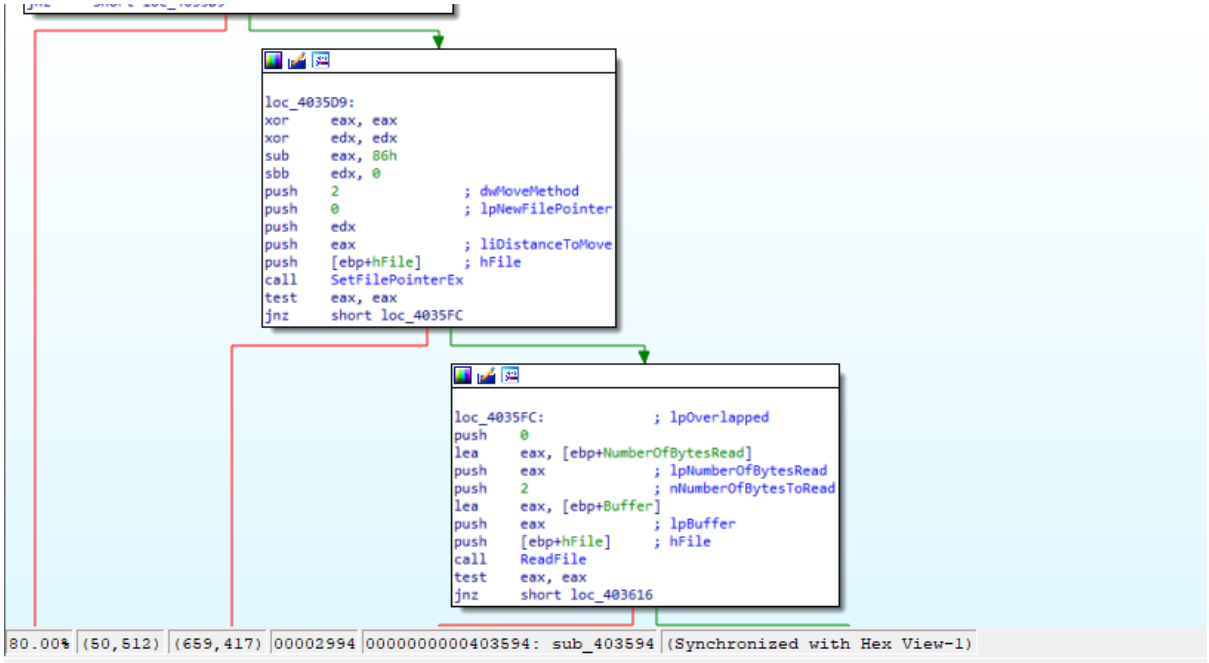
Here are the details of the function calling in the context of files gathering for the setting of the file pointer and the attributes of the files taken into consideration.

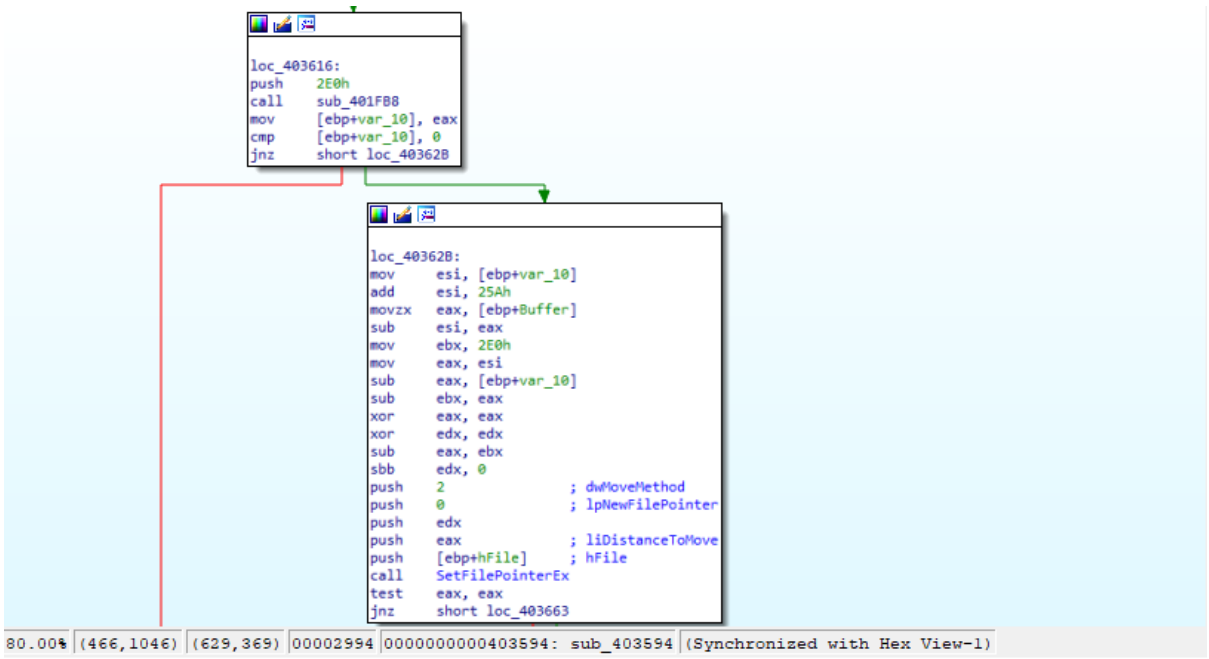

test eax, eax
jz short loc_403EEF
push [ebp+var_258] ; int
push [ebp+var_254] ; int
push esi ; pszPath
call sub_402810
jmp short loc_403EEF
push offset uValue ; lpAddend
call InterlockedIncrement
loc_403EEF:
push esi
call sub_401FF4
jmp short loc_403F0D
loc_403EF7:
lea eax, [ebp+var_8]
push eax ; int
push [ebp+var_258] ; int
push [ebp+var_254] ; int
push esi ; lpExistingFileName
call sub_403970
loc_403F0F:
cmp ds:hFile, 0FFFFFFFF
jz short loc_403F39
100.00% (1144,2538) (791,208) 00003308 0000000000403F08: sub_403CFC+20C (Synchronized with Hex View-1)

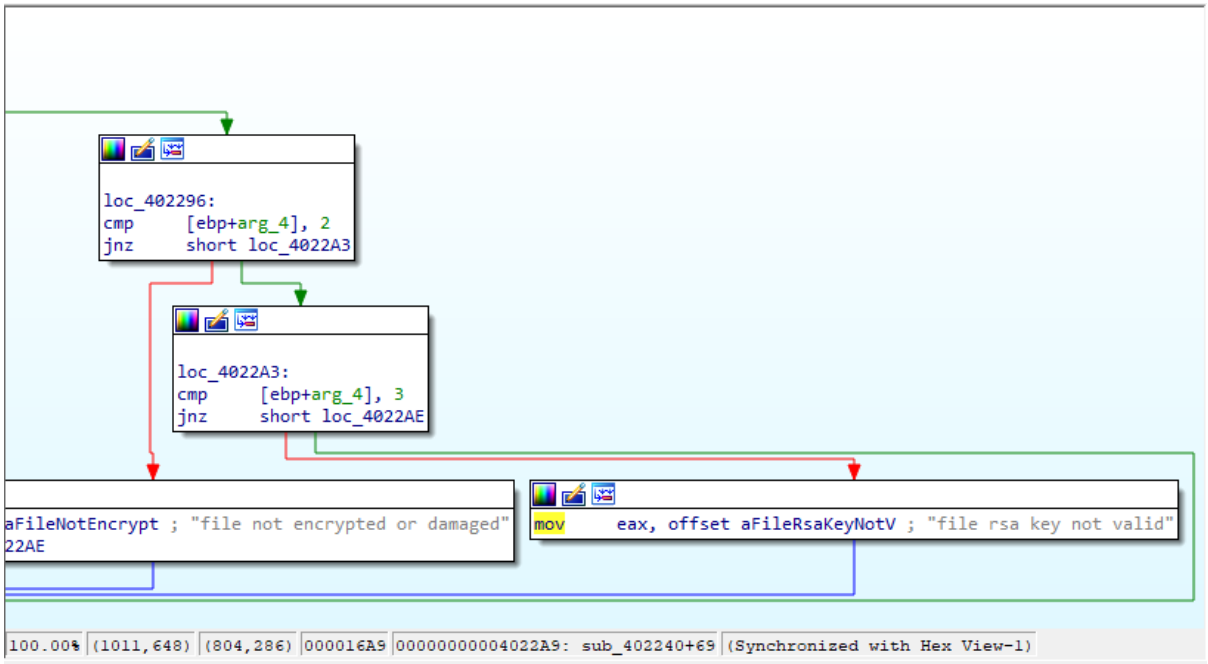

sub_403970 proc near
lpNewFileName= dword ptr -14h
var_10= dword ptr -10h
var_C= dword ptr -0Ch
Handle= dword ptr -8
var_4= dword ptr -4
lpExistingFileName= dword ptr 8
arg_4= dword ptr 0Ch
arg_8= dword ptr 10h
arg_C= dword ptr 14h
push ebp
mov ebp, esp
add esp, 0FFFFFFECh
push ebx
push esi
push edi
mov [ebp+var_4], 0
mov [ebp+var_C], 0
mov [ebp+lpNewFileName], 0
push [ebp+lpExistingFileName] ; lpFileName
call sub_4034DC
test eax, eax
jnz short loc_4039B1
loc_4039B1:
mov edi, [ebp+arg_C]
cmp dword ptr [edi], 0
jz short loc_403A17
80.00% (214,92) (808,290) 00002D70 0000000000403970: sub_403970 (Synchronized with Hex View-1)


cmp [ecx+25Ch], eax
jz short loc_4039D2
push dword ptr [edi]
call sub_401FF4
mov dword ptr [edi], 0
jmp short loc_403A17
loc_403A17:
lea eax, [edi]
push eax ; int
push [ebp+lpExistingFileName] ; lpFileName
call sub_403594
mov [ebp+var_C], eax
cmp dword ptr [edi], 0
jnz short loc_403A2F
loc_4039EC:
mov ebx, [ebp+var_10]
mov ecx, [edi]
push 80h ; Size
lea eax, [ecx+260h]
push eax ; Src
lea eax, [ebx+260h]
80.00% (637,670) (718,412) 00002E1D 0000000000403A1D: sub_403970+AD (Synchronized with Hex View-1)

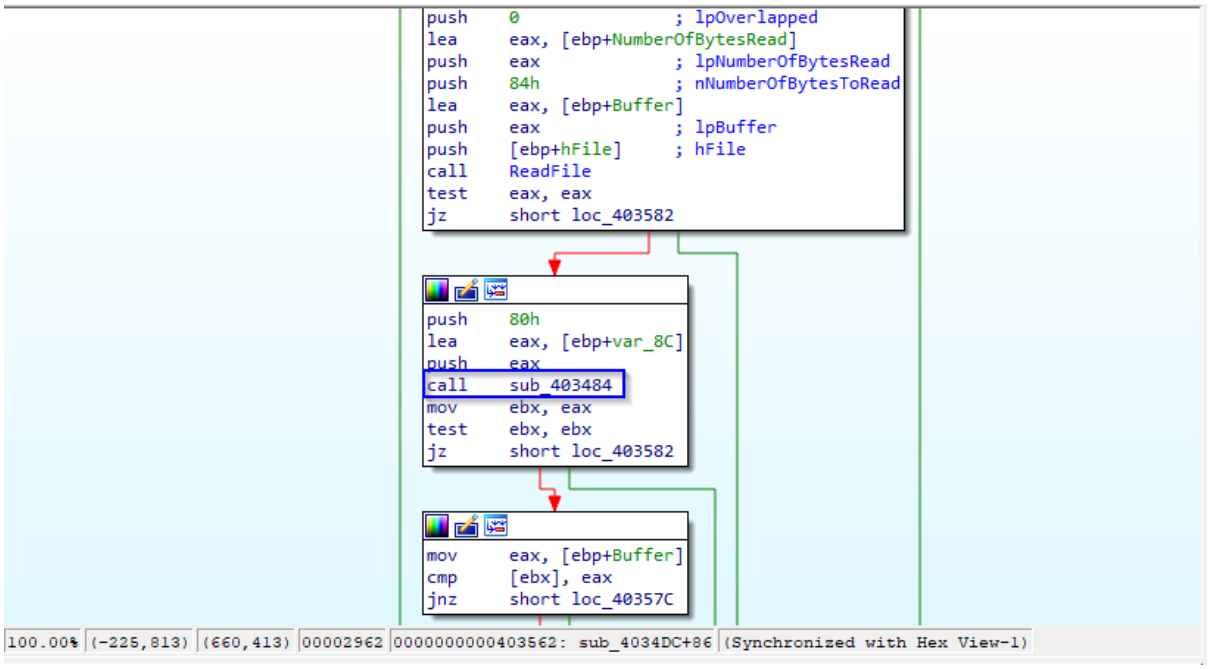

; Attributes: bp-based frame
; int __stdcall sub_403594(LPCWSTR lpFileName, int)
sub_403594 proc near
var_10= dword ptr -10h
Buffer= word ptr -0Ah
NumberOfBytesRead= dword ptr -8
hFile= dword ptr -4
lpFileName= dword ptr 8
arg_4= dword ptr 0Ch
push ebp
mov ebp, esp
add esp, 0FFFFFFF0h
push ebx
push esi
mov [ebp+var_10], 0
mov [ebp+NumberOfBytesRead], 0
mov [ebp+hFile], 0FFFFFFFFh
push 0 ; hTemplateFile
push 80h ; dwFlagsAndAttributes
push 3 ; dwCreationDisposition
push 0 ; lpSecurityAttributes
push 0 ; dwShareMode
push 80000000h ; dwDesiredAccess
push [ebp+lpFileName] ; lpFileName
call CreateFileW
mov [ebp+hFile], eax
cmp [ebp+hFile], 0FFFFFFFFh
jnz short loc_4035D9
80.00% (-32,30) (619,420) 00002994 0000000000403594: sub_403594 (Synchronized with Hex View-1)

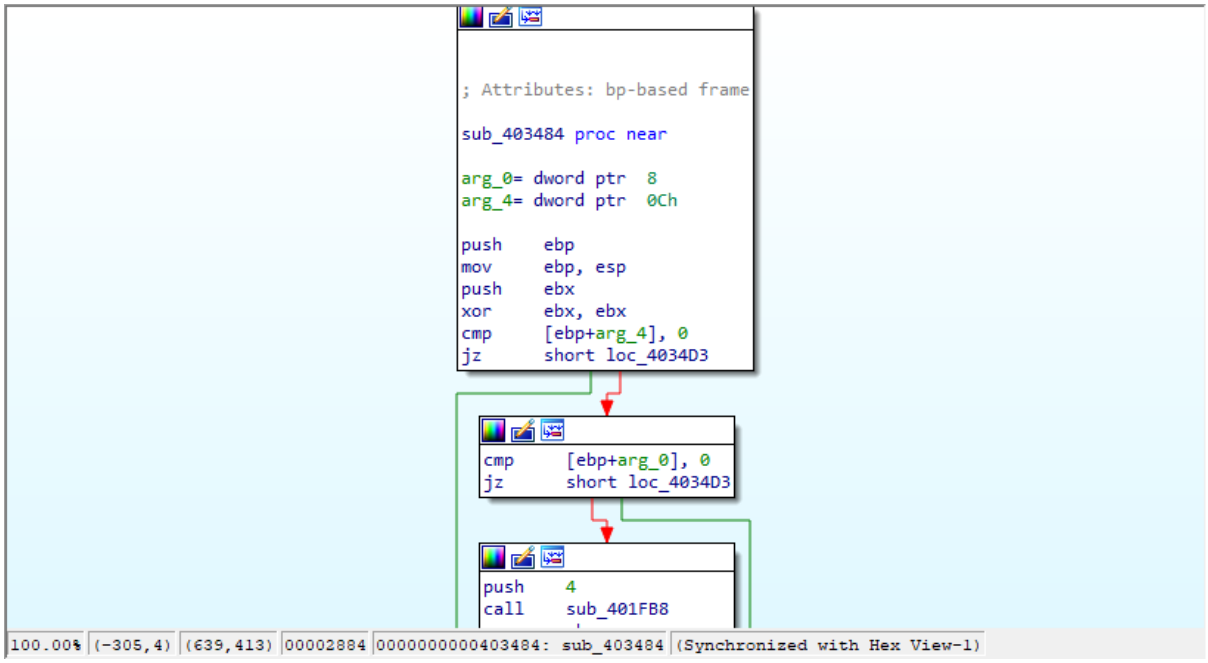

loc_4035D9:
xor eax, eax
xor edx, edx
sub eax, 86h
sbb edx, 0
push 2 ; dwMoveMethod
push 0 ; lpNewFilePointer
push edx
push eax ; liDistanceToMove
push [ebp+hFile] ; hFile
call SetFilePointerEx
test eax, eax
jnz short loc_4035FC
loc_4035FC: ; lpOverlapped
push 0
lea eax, [ebp+NumberOfBytesRead]
push eax ; lpNumberOfBytesRead
push 2 ; nNumberOfBytesToRead
lea eax, [ebp+Buffer]
push eax ; lpBuffer
push [ebp+hFile] ; hFile
call ReadFile
test eax, eax
jnz short loc_403616
80.00% (50,512) (659,417) 00002994 0000000000403594: sub_403594 (Synchronized with Hex View-1)

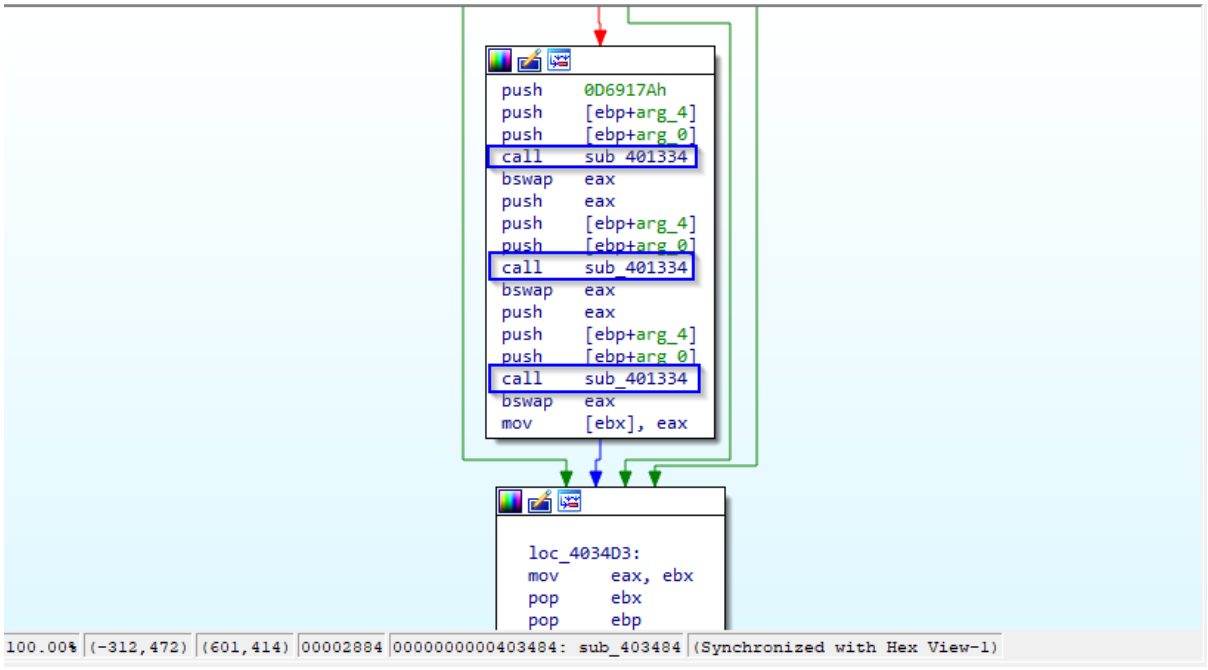

loc_403616:
push 2E0h
call sub_401FB8
mov [ebp+var_10], eax
cmp [ebp+var_10], 0
jnz short loc_40362B
loc_40362B:
mov esi, [ebp+var_10]
add esi, 25Ah
movzx eax, [ebp+Buffer]
sub esi, eax
mov ebx, 2E0h
mov eax, esi
sub eax, [ebp+var_10]
sub ebx, eax
xor eax, eax
xor edx, edx
sub eax, ebx
sbb edx, 0
push 2 ; dwMoveMethod
push 0 ; lpNewFilePointer
push edx
push eax ; liDistanceToMove
push [ebp+hFile] ; hFile
call SetFilePointerEx
test eax, eax
jnz short loc_403663
80.00% (466,1046) (629,369) 00002994 0000000000403594: sub_403594 (Synchronized with Hex View-1)

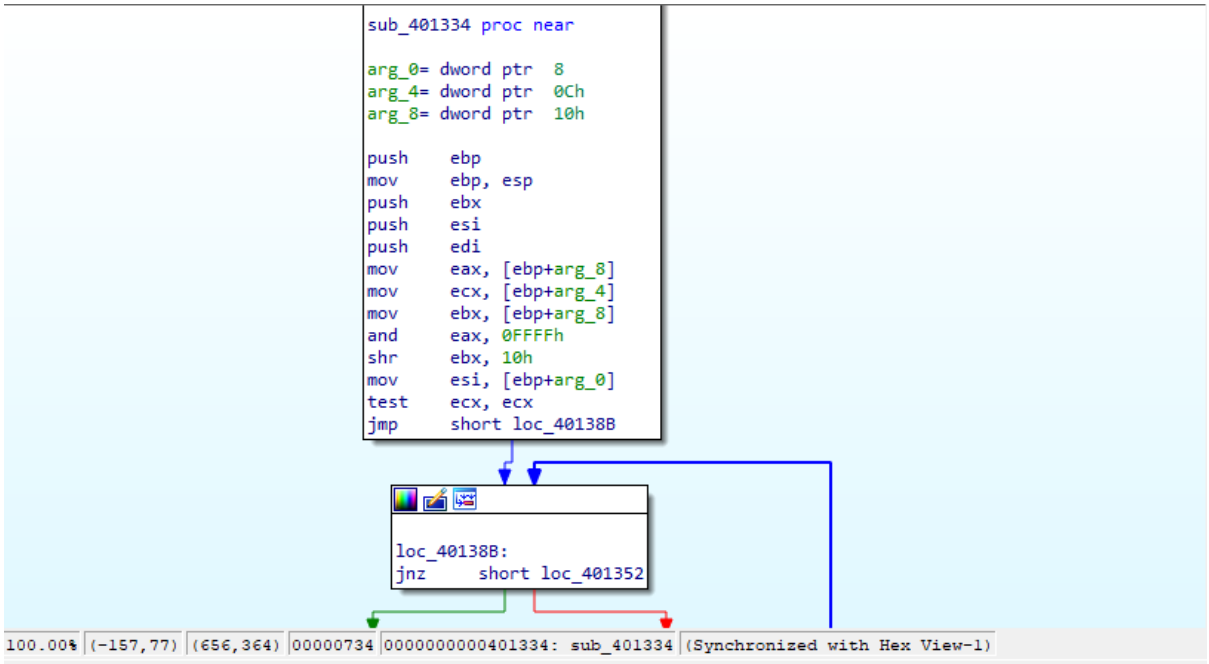


When the encrypted files in input are read, the function **sub_403484** is executed, then a buffer is called and inserted in the eax register. The function sub_403484 contains XOR operations and calls for 3 times (intermittently to bswap operations) the function ***sub_401334***. The ***bswap*** instruction permits to perform the swap of bytes taken in input considered during the execution.

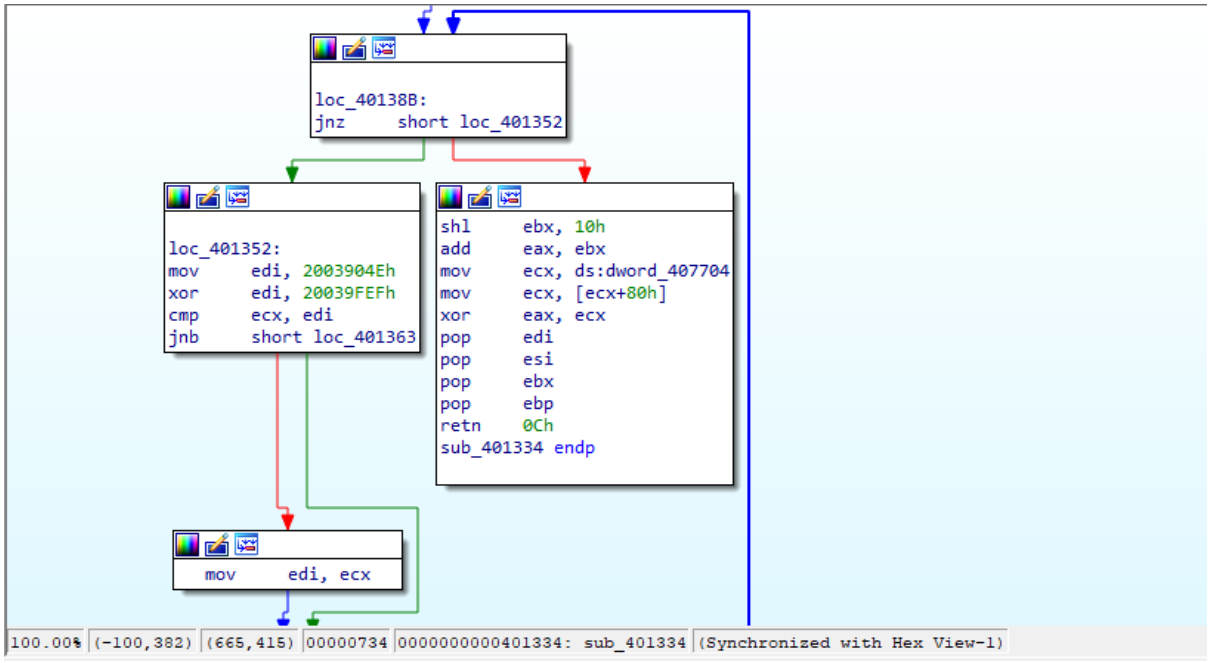

push 0 ; lpOverlapped
lea eax, [ebp+NumberOfBytesRead]
push eax ; lpNumberOfBytesRead
push 84h ; nNumberOfBytesToRead
lea eax, [ebp+Buffer]
push eax ; lpBuffer
push [ebp+hFile] ; hFile
call ReadFile
test eax, eax
jz short loc_403582
push 80h
lea eax, [ebp+var_8C]
push eax
call sub_403484
mov ebx, eax
test ebx, ebx
jz short loc_403582
mov eax, [ebp+Buffer]
cmp [ebx], eax
jnz short loc_40357C
100.00% (-225,813) (660,413) 00002962 0000000000403562: sub_4034DC+86 (Synchronized with Hex View-1)

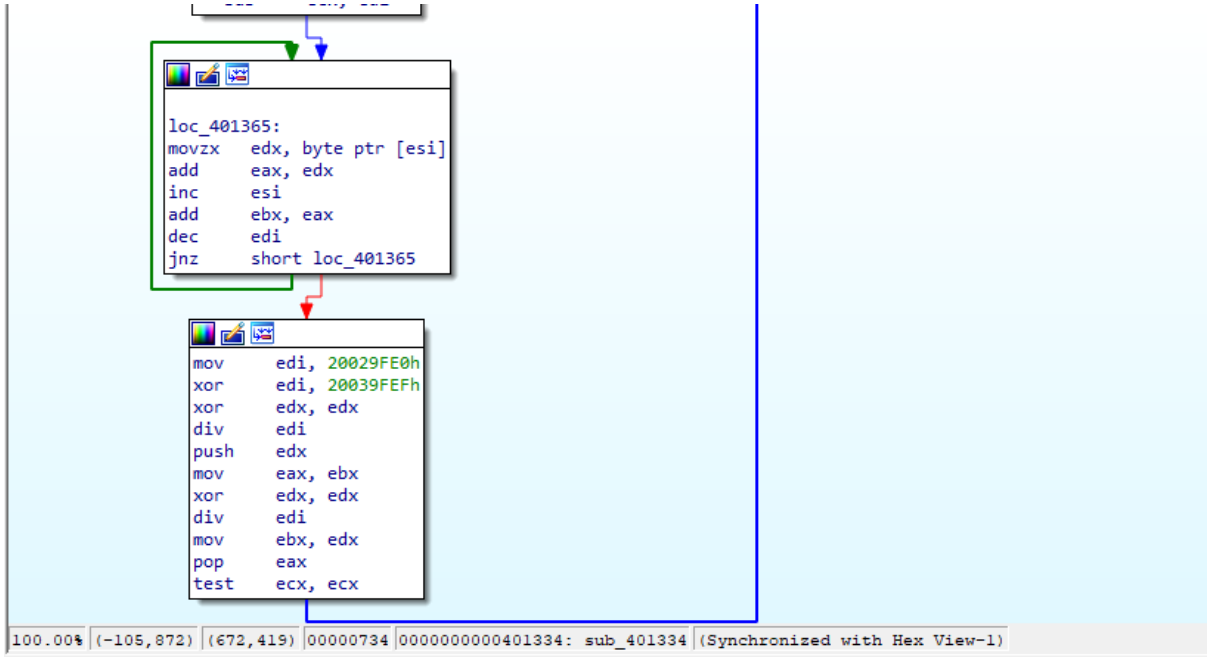

; Attributes: bp-based frame
sub_403484 proc near
arg_0= dword ptr 8
arg_4= dword ptr 0Ch
push ebp
mov ebp, esp
push ebx
xor ebx, ebx
cmp [ebp+arg_4], 0
jz short loc_4034D3
cmp [ebp+arg_0], 0
jz short loc_4034D3
push 4
call sub_401FB8
100.00% (-305,4) (639,413) 00002884 0000000000403484: sub_403484 (Synchronized with Hex View-1)

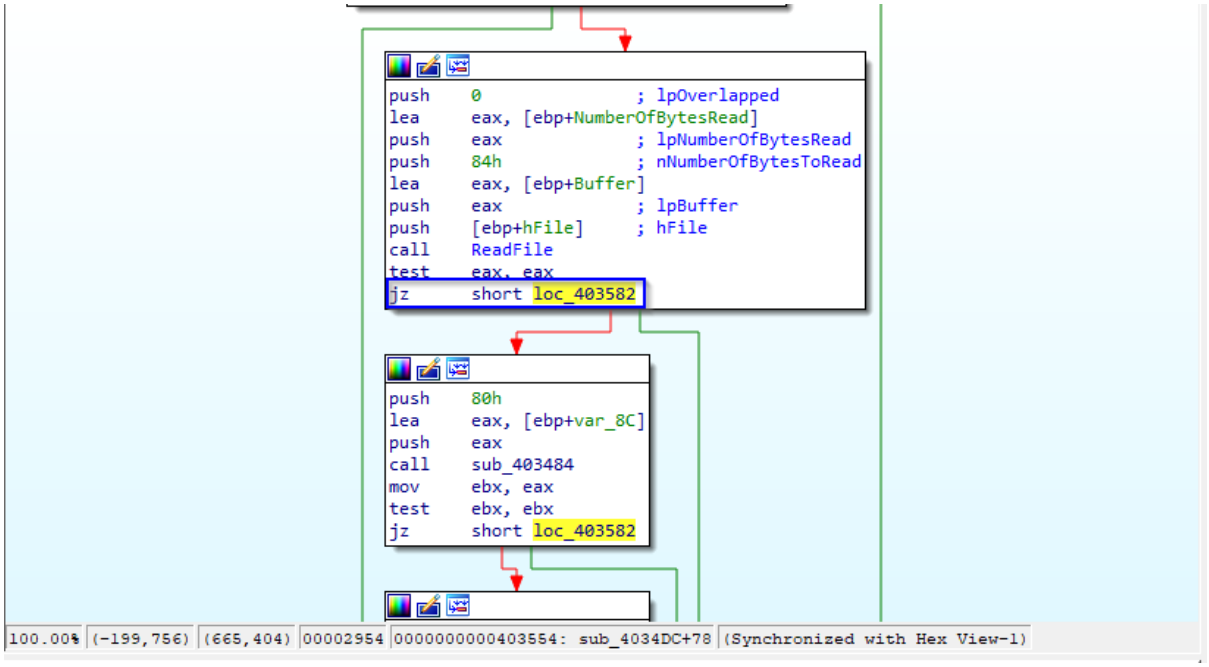


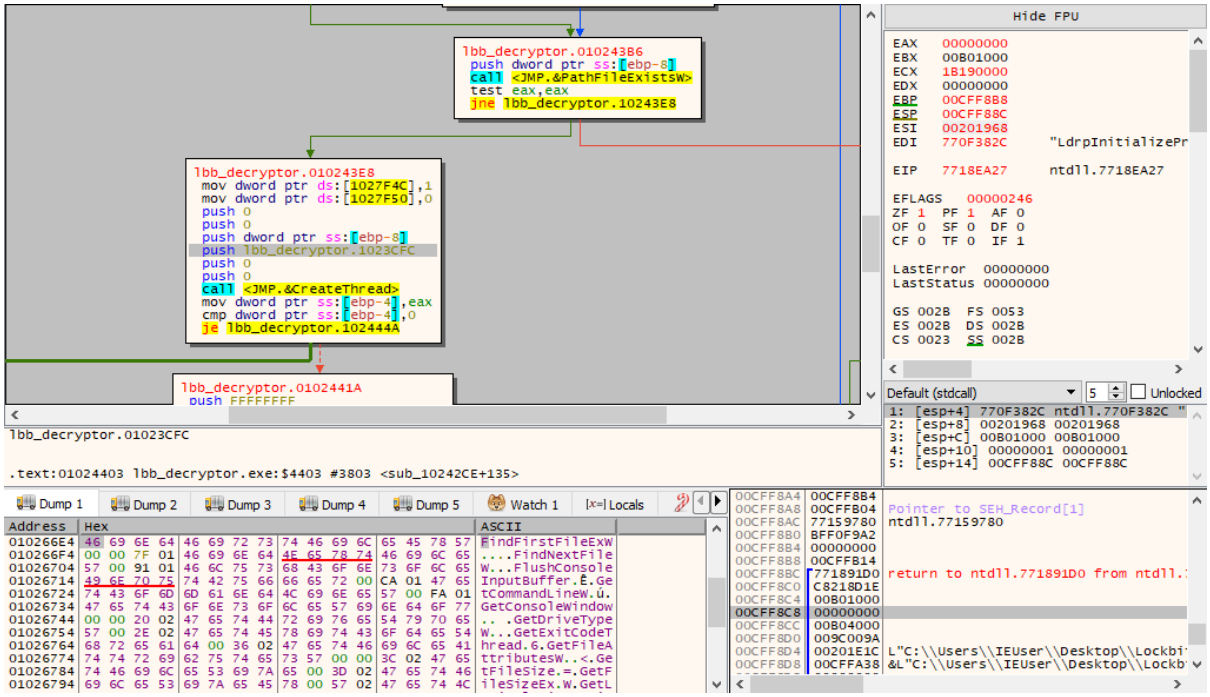


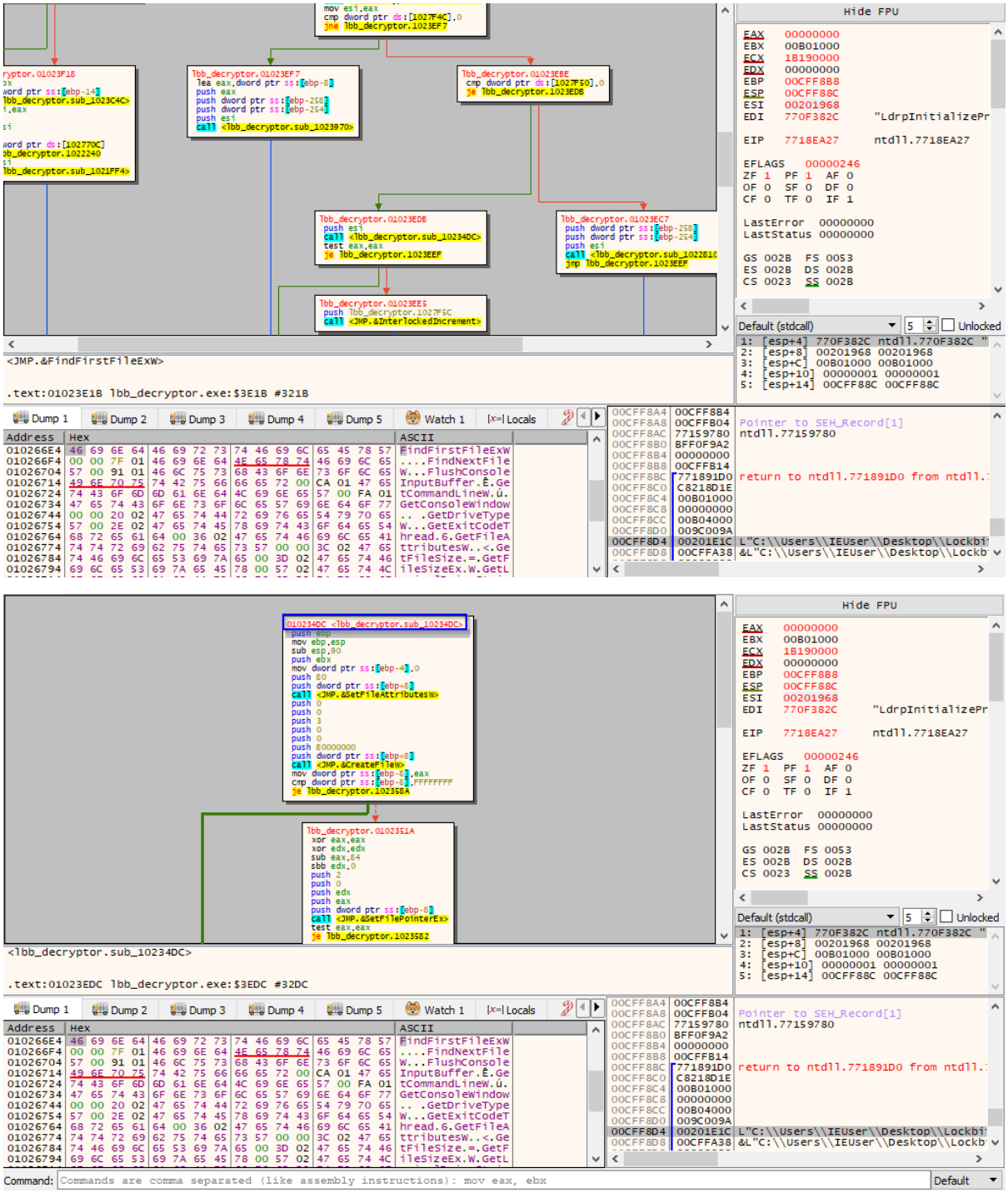
Follows, instead, the execution of the **XOR** instructions for the value ***20039FEFh:***


loc_40138B:
jnz short loc_401352
loc_401352:
mov edi, 2003904Eh
xor edi, 20039FEFh
cmp ecx, edi
jnb short loc_401363
shl ebx, 10h
add eax, ebx
mov ecx, ds:dword_407704
mov ecx, [ecx+80h]
xor eax, ecx
pop edi
pop esi
pop ebx
pop ebp
retn 0Ch
sub_401334 endp
mov edi, ecx
100.00% (-100,382) (665,415) 00000734 0000000000401334: sub_401334 (Synchronized with Hex View-1)

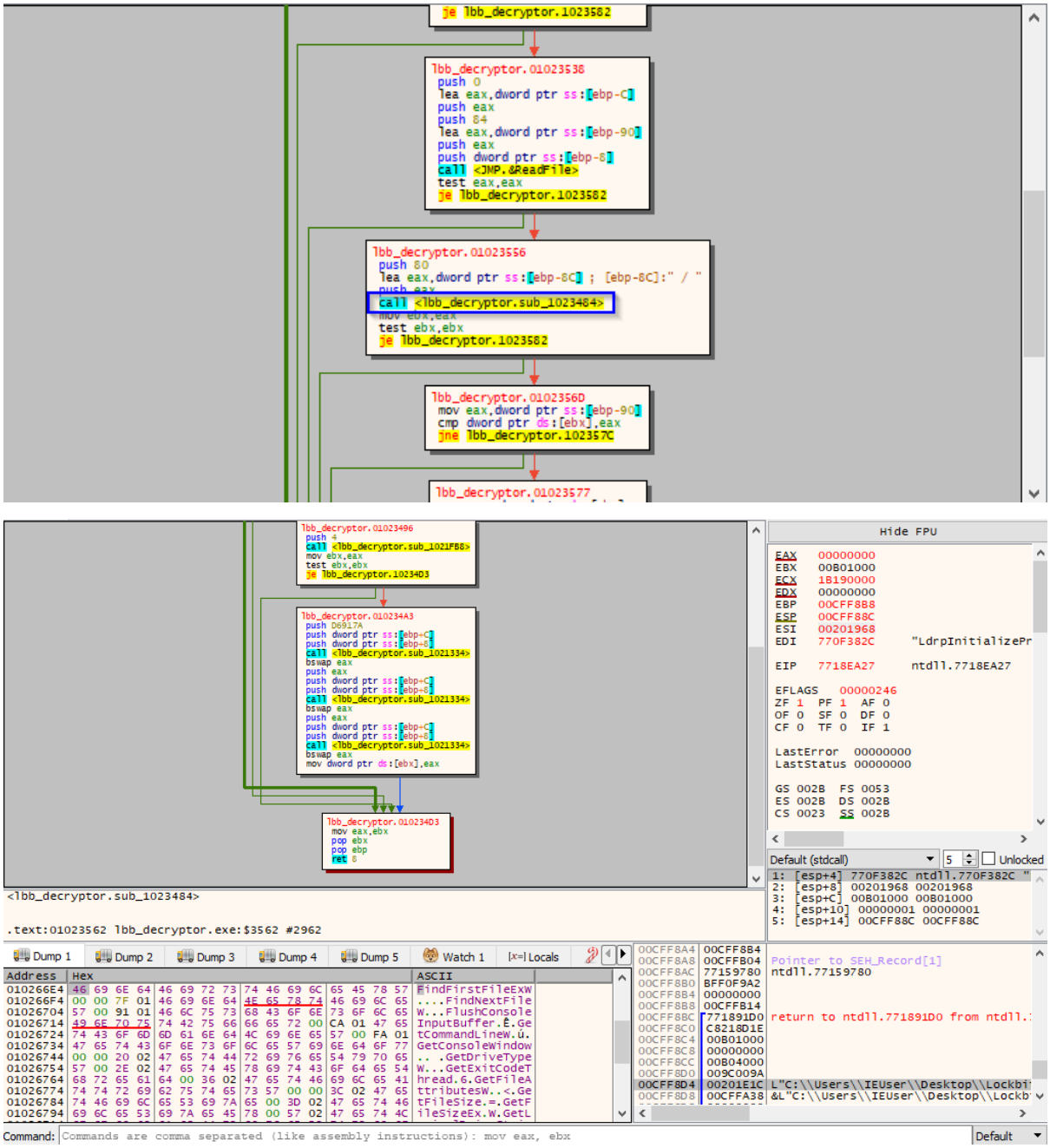

loc_401365:
movzx edx, byte ptr [esi]
add eax, edx
inc esi
add ebx, eax
dec edi
jnz short loc_401365
mov edi, 20029FE0h
xor edi, 20039FEFh
xor edx, edx
div edi
push edx
mov eax, ebx
xor edx, edx
div edi
mov ebx, edx
pop eax
test ecx, ecx
100.00% (-105,872) (672,419) 00000734 0000000000401334: sub_401334 (Synchronized with Hex View-1)


push 0 ; lpOverlapped
lea eax, [ebp+NumberOfBytesRead]
push eax ; lpNumberOfBytesRead
push 84h ; nNumberOfBytesToRead
lea eax, [ebp+Buffer]
push eax ; lpBuffer
push [ebp+hFile] ; hFile
call ReadFile
test eax, eax
jz short loc_403582
push 80h
lea eax, [ebp+var_8C]
push eax
call sub_403484
mov ebx, eax
test ebx, ebx
jz short loc_403582
100.00% (-199,756) (665,404) 00002954 0000000000403554: sub_4034DC+78 (Synchronized with Hex View-1)


lbb_decryptor.010243B6
push dword ptr ss:[ebp-8]
call <JMP.&PathFileExistsW>
test eax,eax
jne lbb_decryptor.10243E8
lbb_decryptor.010243E8
mov dword ptr ds:[1027F4C],1
mov dword ptr ds:[1027F50],0
push 0
push 0
push dword ptr ss:[ebp-8]
push lbb_decryptor.1023CFC
push 0
push 0
call <JMP.&CreateThread>
mov dword ptr ss:[ebp-4],eax
cmp dword ptr ss:[ebp-4],0
je lbb_decryptor.102444A
lbb_decryptor.0102441A
push FFFFFFFF
Hide FPU
EAX 00000000
EBX 00B01000
ECX 1B190000
EDX 00000000
EBP 00CFF8B8
ESP 00CFF88C
ESI 00201968
EDI 770F382C "LdrpInitializePr
EIP 7718EA27 ntdll.7718EA27
EFLAGS 00000246
ZF 1 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1
LastError 00000000
LastStatus 00000000
GS 002B FS 0053
ES 002B DS 002B
CS 0023 SS 002B
Default (stdcall)
1: [esp+4] 770F382C ntdll.770F382C
2: [esp+8] 00201968 00201968
3: [esp+C] 00B01000 00B01000
4: [esp+10] 00000001 00000001
5: [esp+14] 00CFF88C 00CFF88C
lbb_decryptor.01023CFC
.text:01024403 lbb_decryptor.exe:$4403 #3803 <sub_10242CE+135>
Dump 1
Dump 2
Dump 3
Dump 4
Dump 5
Watch 1
[x=] Locals
Address Hex ASCII
010266E4 46 69 6E 64 46 69 72 73 74 46 69 6C 65 45 78 57 FindFirstFileExW
010266F4 00 00 7F 01 46 69 6E 64 4E 65 78 74 46 69 6C 65 ....FindNextFile
01026704 57 00 91 01 46 6C 75 73 68 43 6F 6E 73 6F 6C 65 W...FlushConsole
01026714 49 6E 70 75 74 42 75 66 66 65 72 00 CA 01 47 65 InputBuffer.Ê.Ge
01026724 74 43 6F 6D 6D 61 6E 64 4C 69 6E 65 57 00 FA 01 tCommandLineW.ú.
01026734 47 65 74 43 6F 6E 73 6F 6C 65 57 69 6E 64 6F 77 GetConsoleWindow
01026744 00 00 20 02 47 65 74 44 72 69 76 65 54 79 70 65 .. .GetDriveType
01026754 57 00 2E 02 47 65 74 45 78 69 74 43 6F 64 65 54 W...GetExitCodeT
01026764 68 72 65 61 64 00 36 02 47 65 74 46 69 6C 65 41 hread.6.GetFileA
01026774 74 74 72 69 62 75 74 65 73 57 00 00 3C 02 47 65 ttributesW..<.Ge
01026784 74 46 69 6C 65 53 69 7A 65 00 3D 02 47 65 74 46 tFileSize.=.GetF
01026794 69 6C 65 53 69 7A 65 45 78 00 57 02 47 65 74 4C ileSizeEx.W.GetL
00CFF8A4 00CFF8B4
00CFF8A8 00CFFB04 Pointer to SEH_Record[1]
00CFF8AC 77159780 ntdll.77159780
00CFF8B0 BFF0F9A2
00CFF8B4 00000000
00CFF8B8 00CFFB14
00CFF8BC 771891D0 return to ntdll.771891D0 from ntdll.
00CFF8C0 C8218D1E
00CFF8C4 00B01000
00CFF8C8 00000000
00CFF8CC 00B04000
00CFF8D0 009C009A
00CFF8D4 00201E1C L"C:\\Users\\IEUser\\Desktop\\Lockbi
00CFF8D8 00CFFA38 &L"C:\\Users\\IEUser\\Desktop\\Lockb

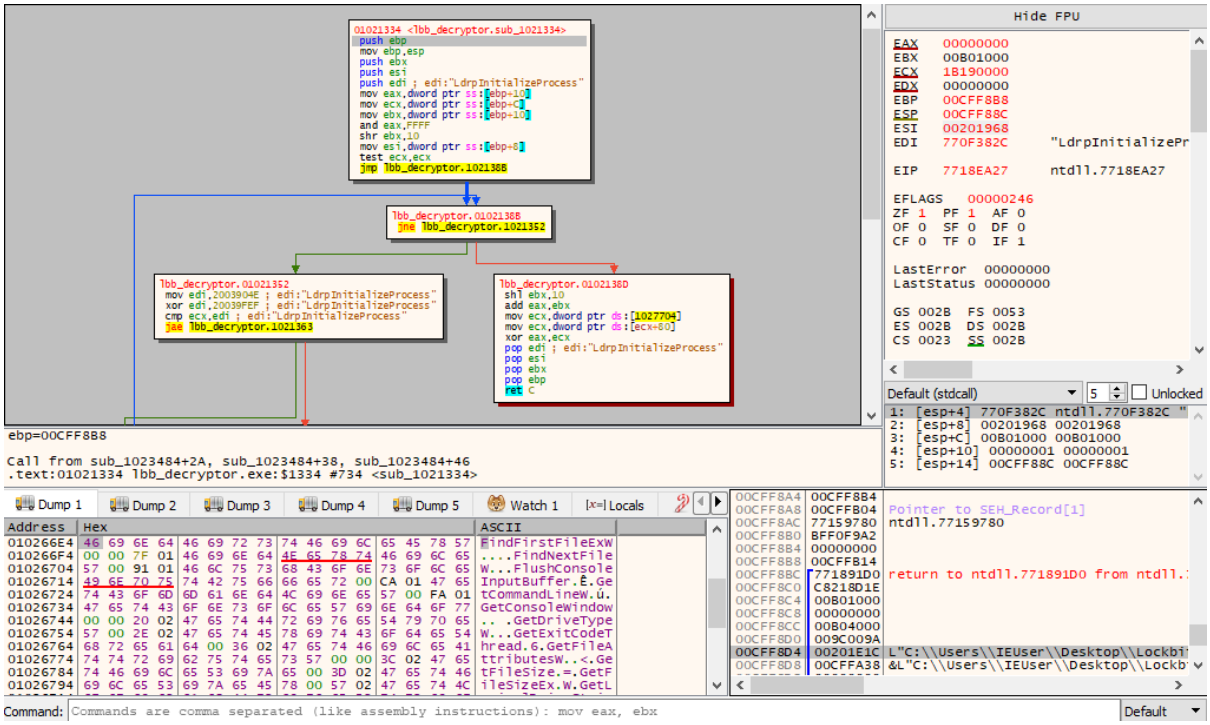


Following is the identification of the same execution context in debugging environment:



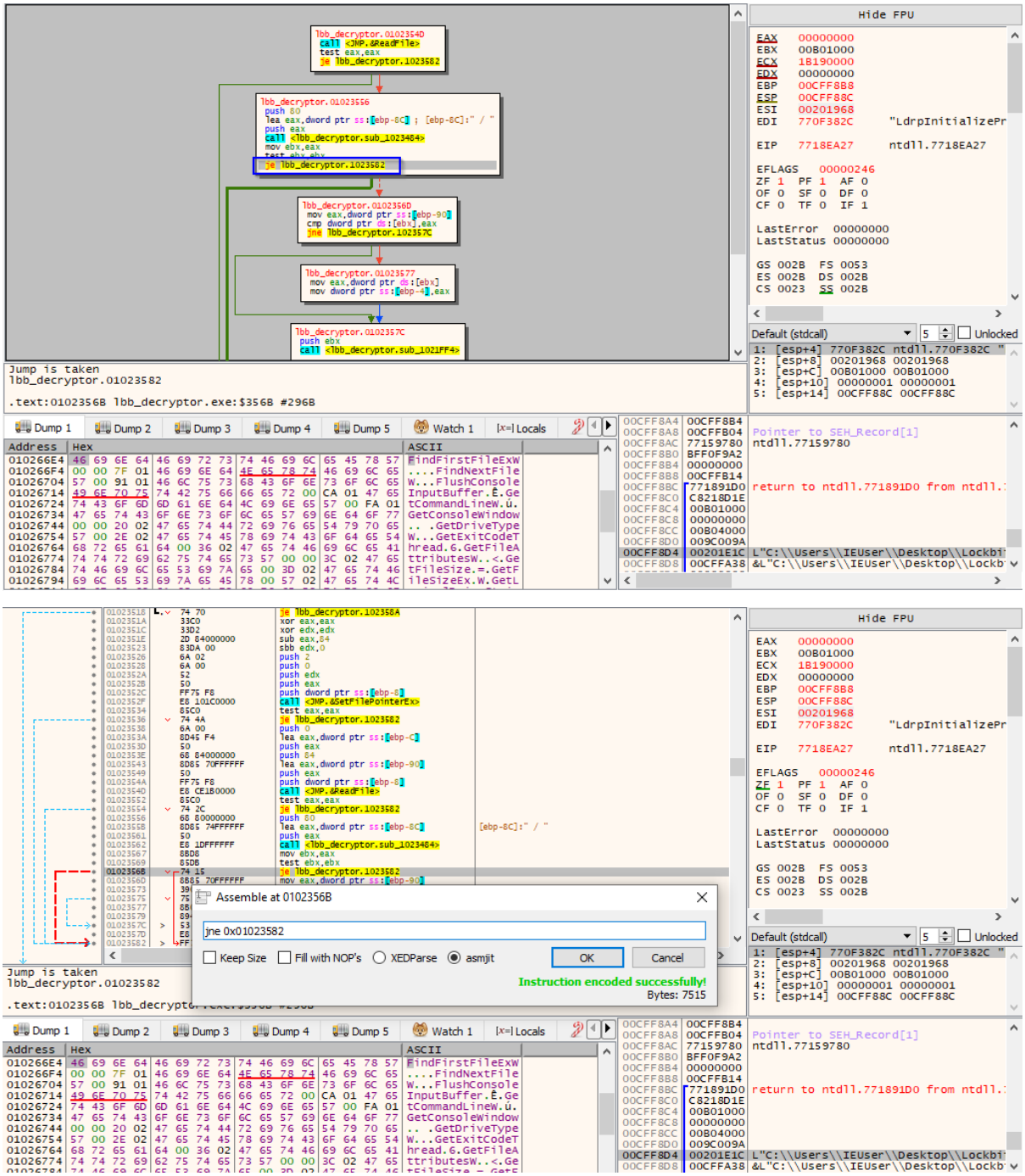
By decompiling in native C++ is possible to have the evidence of bit shifting performed on the attributes of the files taken in input:

```
uint32_t* fun_1021fb8(uint32_t a1);

uint32_t fun_1021334(uint32_t a1, uint32_t a2, uint32_t a3, uint32_t a4, uint32_t a5, uint32_t a6, uint32_t a7, uint32_t a8, uint32_t a9, uint32_t a10);

uint32_t* fun_1023484(uint32_t a1, uint32_t a2, uint32_t a3) {
  uint32_t* ebx4;
  uint32_t* eax5;
  uint32_t ebx6;
  uint32_t ebp7;
  uint32_t eax8;
  uint32_t v9;
  uint32_t eax10;
  uint32_t eax11;

  ebx4 = (uint32_t*)0;
  if (a2 && (a1 && (eax5 = fun_1021fb8(4), ebx4 = eax5, !!ebx4))) {
    eax8 = fun_1021334(a1, a2, 0xd6917a, 4, ebx6, ebp7, __return_address(), a1, a2, a3);
    v9 = eax8 >> 24 | eax8 >> 8 & 0xff00 | eax8 << 8 & 0xff0000 | eax8 << 24;
    eax10 = fun_1021334(a1, a2, v9, a1, a2, 0xd6917a, 4, ebx6, ebp7, __return_address());
    eax11 = fun_1021334(a1, a2, eax10 >> 24 | eax10 >> 8 & 0xff00 | eax10 << 8 & 0xff0000 | eax10 << 24, a1, a2, v9, a1, a2, 0xd6917a, 4);
    *ebx4 = eax11 >> 24 | eax11 >> 8 & 0xff00 | eax11 << 8 & 0xff0000 | eax11 << 24;
  }
  return ebx4;
```

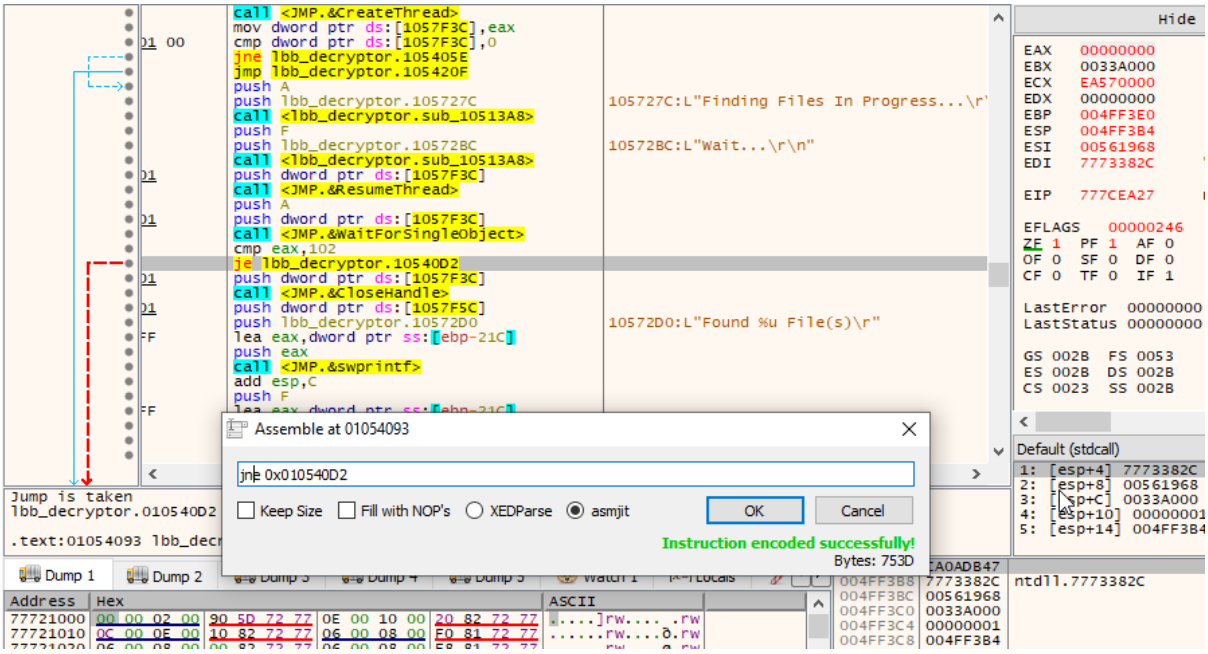


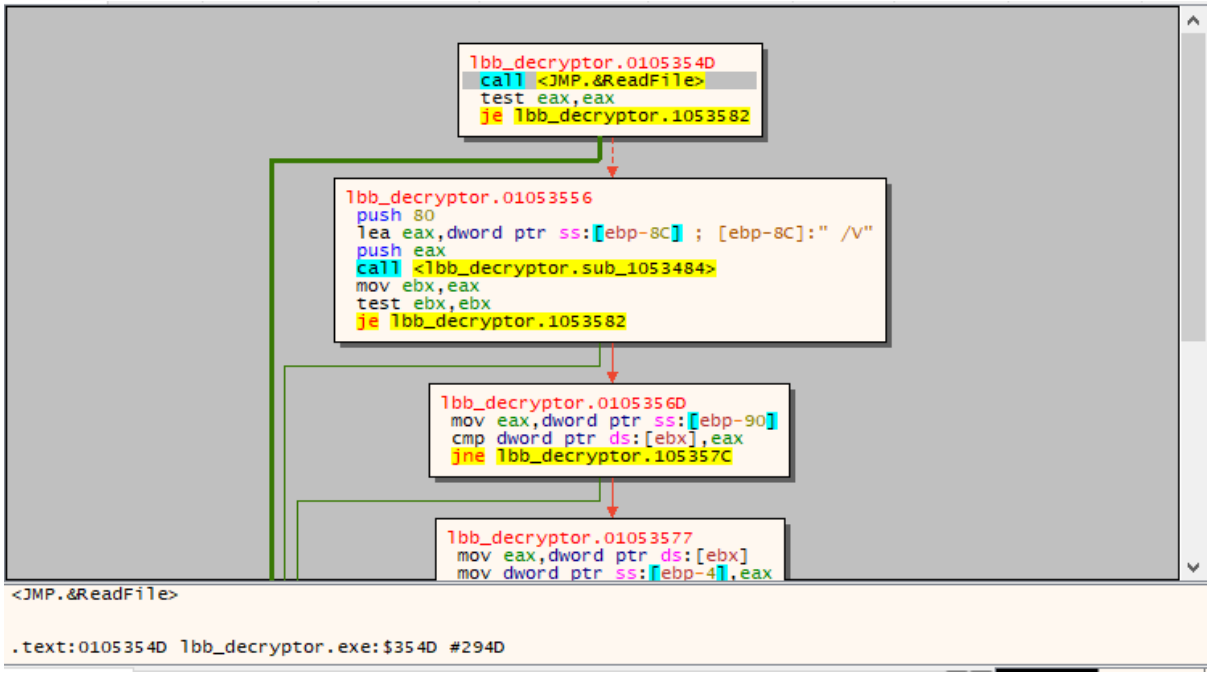
```
struct s0 {
    int8_t[128] pad128;
    uint32_t f128;
};

struct s0* g1027704 = (struct s0*)0;

uint32_t fun_1021334(uint8_t* a1, uint32_t a2, uint32_t a3) {
    uint32_t ecx4;
    uint32_t eax5;
    uint32_t ebx6;
    uint8_t* esi7;
    int1_t zf8;
    uint32_t edi9;
    struct s0* ecx10;

    ecx4 = a2;
    eax5 = a3 & 0xffff;
    ebx6 = a3 >> 16;
    esi7 = a1;
    zf8 = ecx4 == 0;
    while (!zf8) {
        edi9 = 0xfa1;
        if (ecx4 < 0xfa1) {
            edi9 = ecx4;
        }
        ecx4 = ecx4 - edi9;
        do {
            eax5 = eax5 + *esi7;
            ++esi7;
            ebx6 = ebx6 + eax5;
            --edi9;
        } while (edi9);
        ebx6 = ebx6 % 0x1000f;
        eax5 = eax5 % 0x1000f;
        zf8 = ecx4 == 0;
    }
    ecx10 = g1027704;
    return eax5 + (ebx6 << 16) ^ ecx10->f128;
}
```

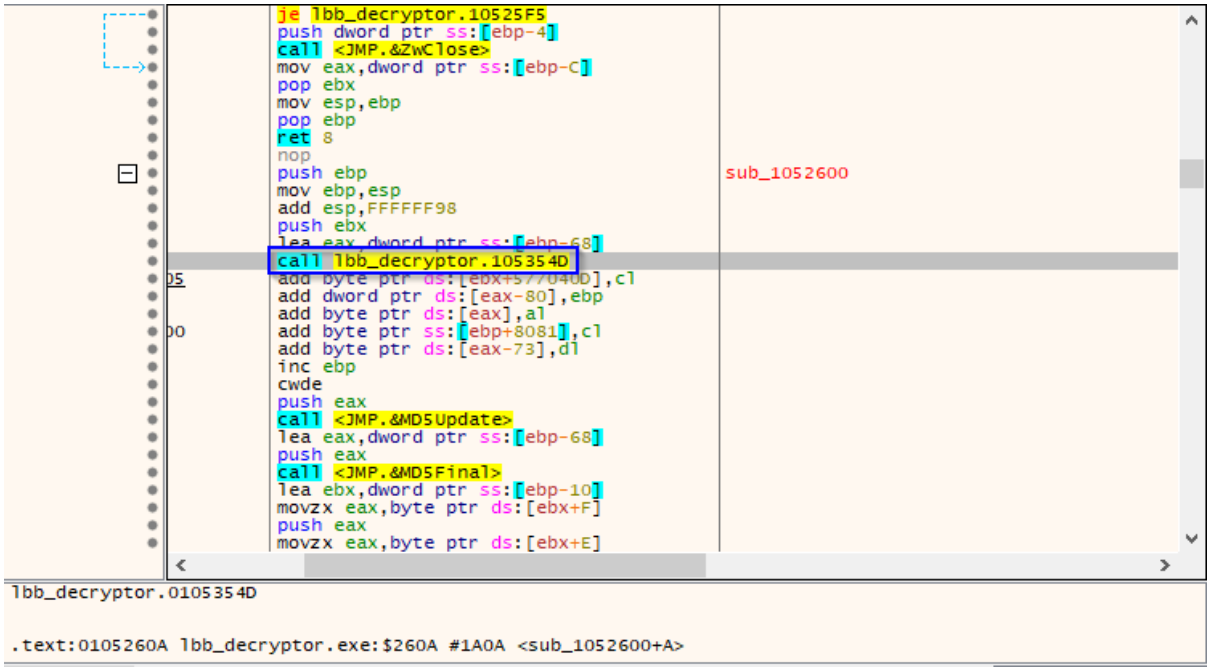
Here's an example of assembly modification related to the execution context in which is performed the call to XOR and bitswapping function.



Here are the details of an attempt to modify a jump instruction in the context of files gathering to **redirect** the execution to the ***modification of attributes*** of the encrypted files:


call <JMP.&CreateThread>
mov dword ptr ds:[1057F3C],eax
cmp dword ptr ds:[1057F3C],0
jne lbb_decryptor.105405E
jmp lbb_decryptor.105420F
push A
push lbb_decryptor.105727C
105727C:L"Finding Files In Progress...\r"
call <lbb_decryptor.sub_10513A8>
push F
push lbb_decryptor.10572BC
10572BC:L"Wait...\r\n"
call <lbb_decryptor.sub_10513A8>
push dword ptr ds:[1057F3C]
call <JMP.&ResumeThread>
push A
push dword ptr ds:[1057F3C]
call <JMP.&WaitForSingleObject>
cmp eax,102
je lbb_decryptor.10540D2
push dword ptr ds:[1057F3C]
call <JMP.&CloseHandle>
push dword ptr ds:[1057F5C]
push lbb_decryptor.10572D0
10572D0:L"Found %u File(s)\r"
lea eax,dword ptr ss:[ebp-21C]
push eax
call <JMP.&swprintf>
add esp,C
push F
Assemble at 01054093
jne 0x010540D2
Keep Size
Fill with NOP's
XEDParse
asmjit
OK
Cancel
Instruction encoded successfully!
Bytes: 753D
Jump is taken
lbb_decryptor.010540D2
.text:01054093 lbb_decr
Hide
EAX 00000000
EBX 0033A000
ECX EA570000
EDX 00000000
EBP 004FF3E0
ESP 004FF3B4
ESI 00561968
EDI 7773382C
EIP 777CEA27
EFLAGS 00000246
ZF 1 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1
LastError 00000000
LastStatus 00000000
GS 002B FS 0053
ES 002B DS 002B
CS 0023 SS 002B
Default (stdcall)
1: [esp+4] 7773382C
2: [esp+8] 00561968
3: [esp+C] 0033A000
4: [esp+10] 00000001
5: [esp+14] 004FF3B4
Dump 1
Dump 2
Address
Hex
ASCII
77721000 00 00 02 00 90 5D 72 77 0E 00 10 00 20 82 72 77
77721010 0C 00 0E 00 10 82 72 77 06 00 08 00 F0 81 72 77
004FF3B8 7773382C
004FF3BC 00561968
004FF3C0 0033A000
004FF3C4 00000001
004FF3C8 004FF3B4
ntdll.7773382C

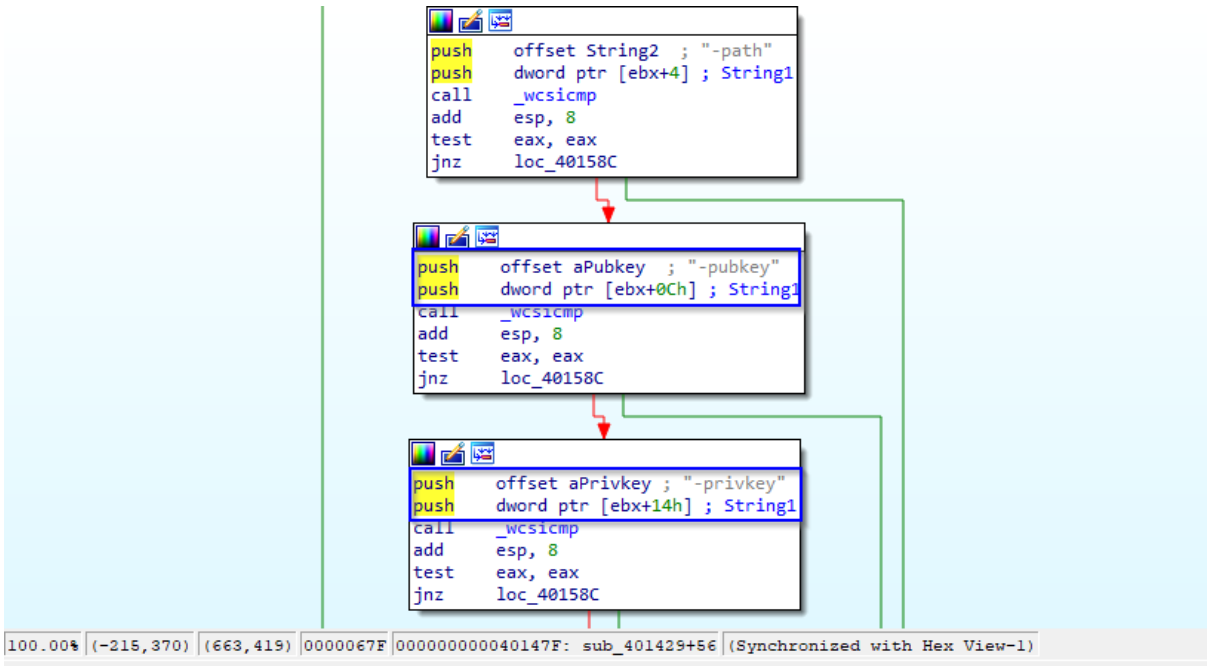

lbb_decryptor.0105354D
call <JMP.&ReadFile>
test eax,eax
je lbb_decryptor.1053582
lbb_decryptor.01053556
push 80
lea eax,dword ptr ss:[ebp-8C] ; [ebp-8C]:" /V"
push eax
call <lbb_decryptor.sub_1053484>
mov ebx,eax
test ebx,ebx
je lbb_decryptor.1053582
lbb_decryptor.0105356D
mov eax,dword ptr ss:[ebp-90]
cmp dword ptr ds:[ebx],eax
jne lbb_decryptor.105357C
lbb_decryptor.01053577
mov eax,dword ptr ds:[ebx]
mov dword ptr ss:[ebp-4],eax
<JMP.&ReadFile>
.text:0105354D lbb_decryptor.exe:$354D #294D



To generate the couple of public and private keys (where the public key is used by the ransomware and the private key by the decryptor) is called the executable **keygen.exe**, which creates the variables of the couple of keys respectively as follows:

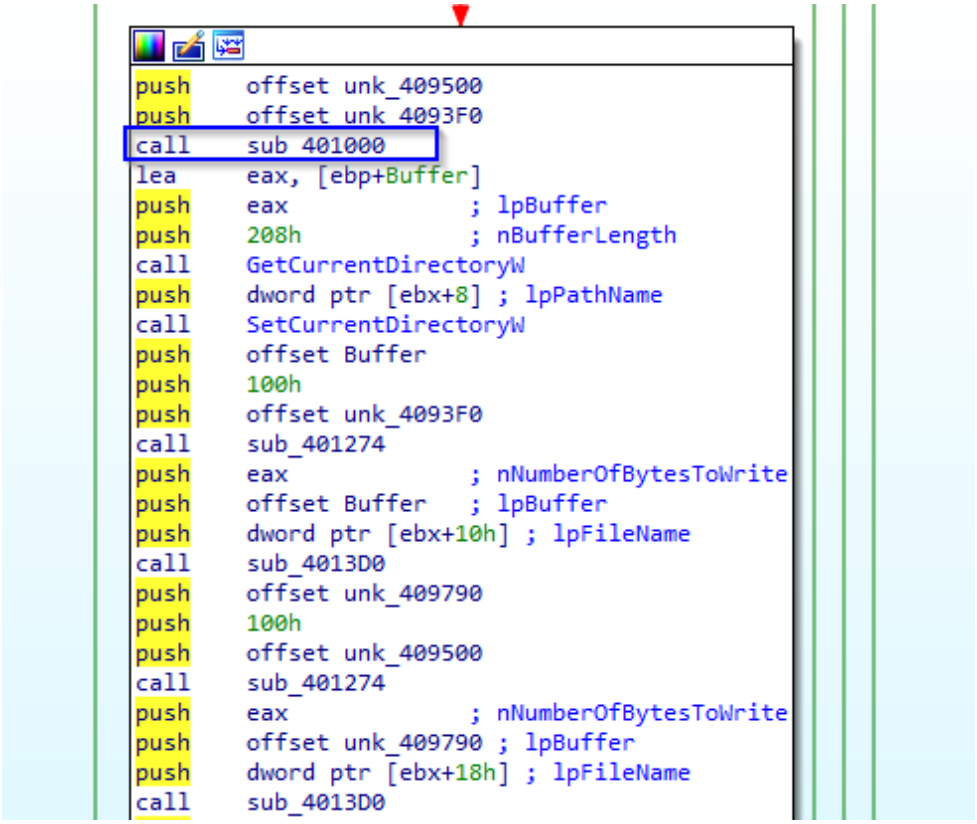
Public key -> ***ebx+14h***

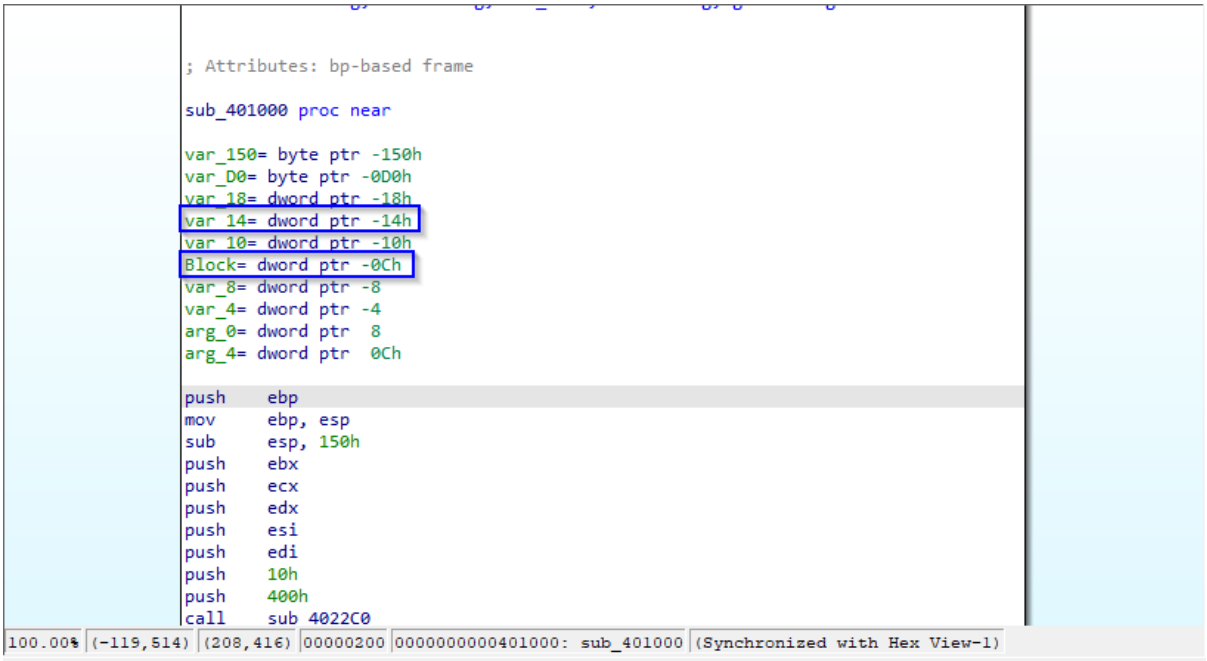
Private key -> ***ebx+0Ch***

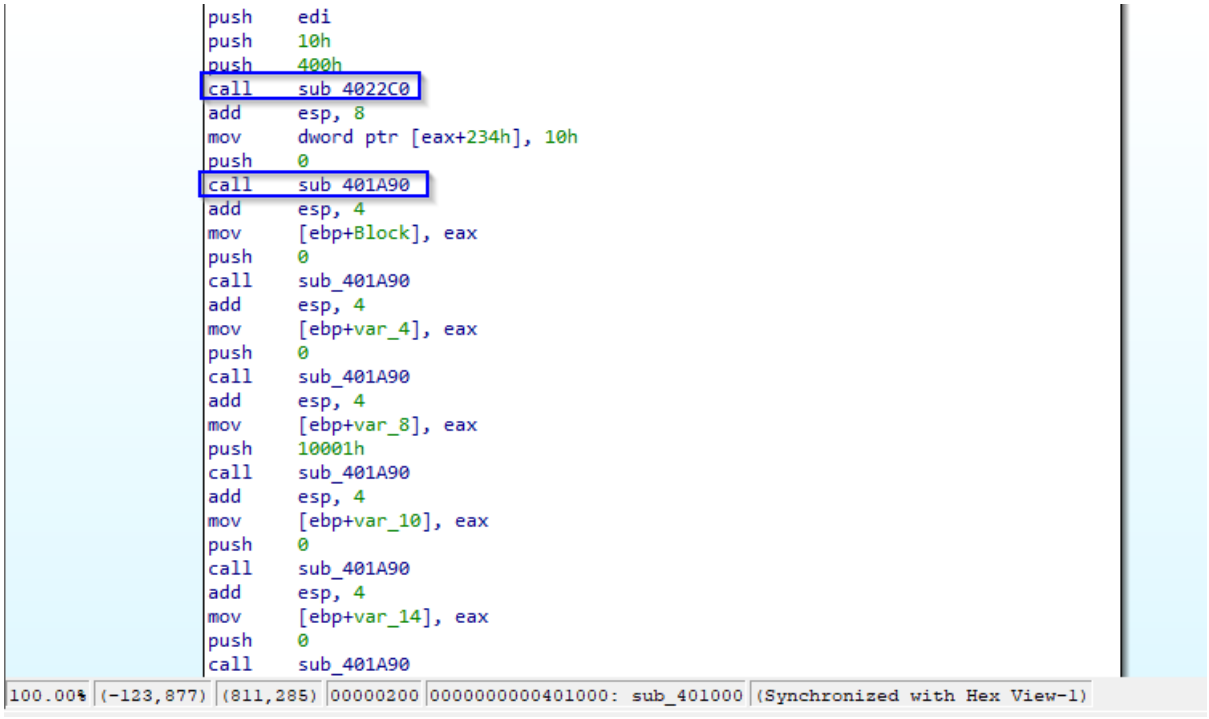


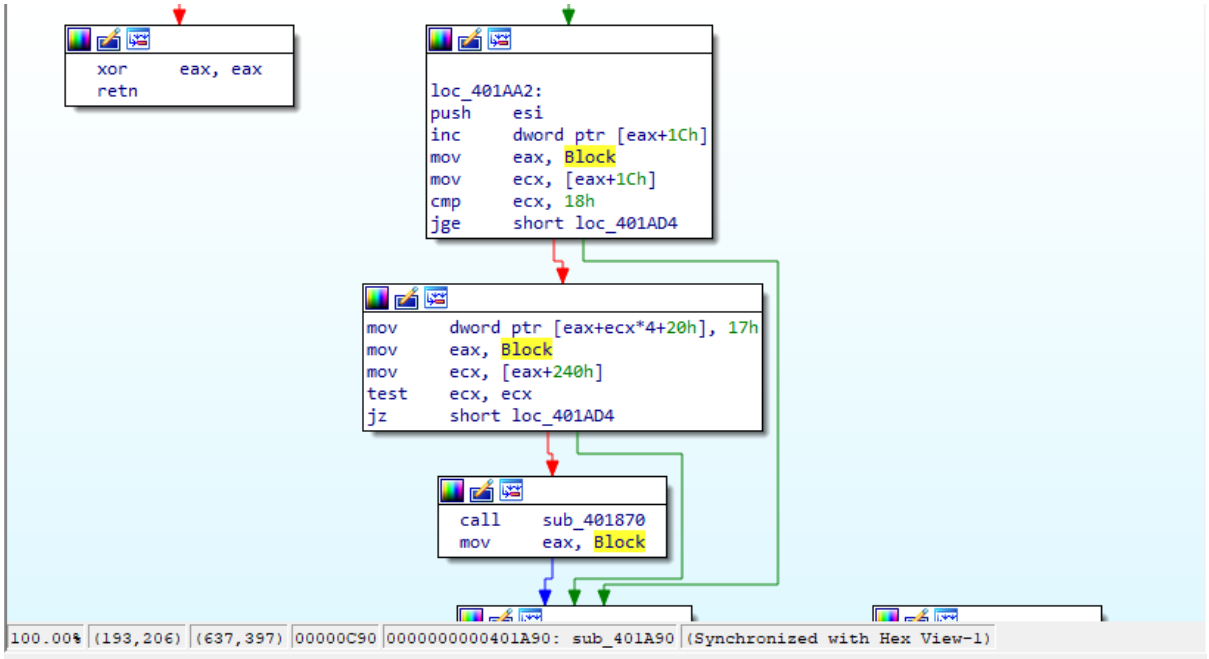
Subsequently it is called the function sub_401000 to call the execution of the cryptography

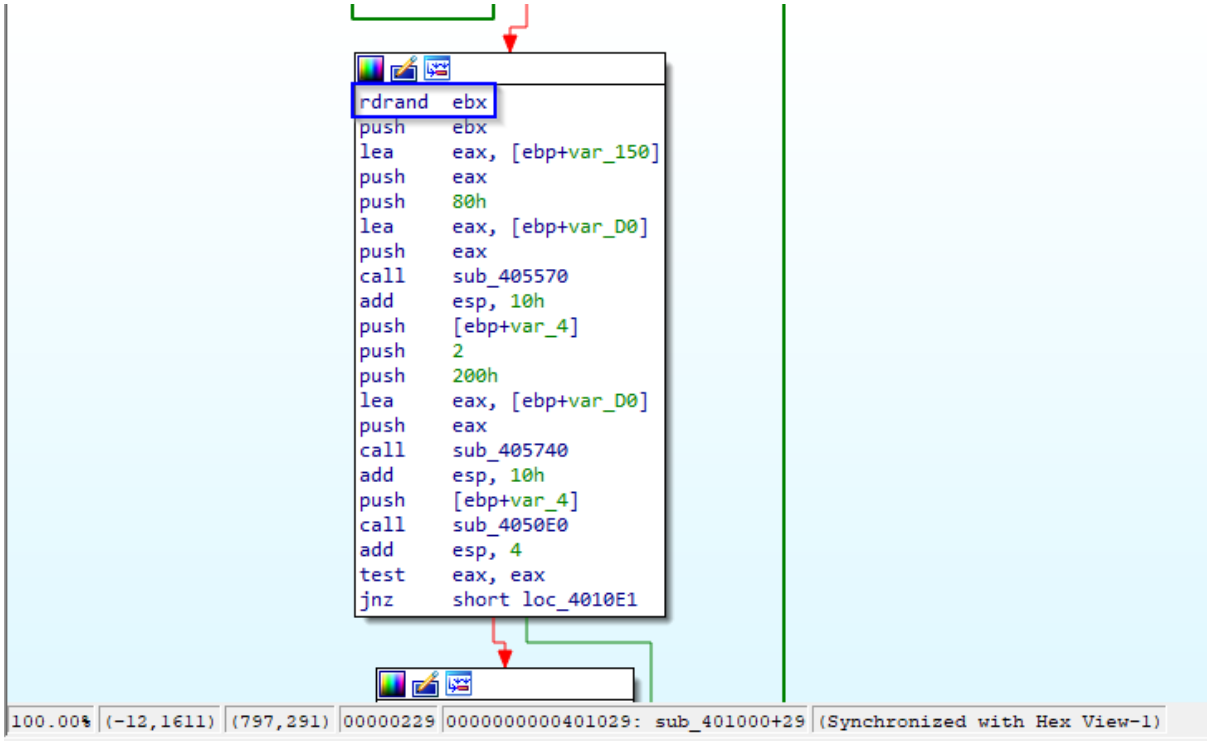
library and the randomic generation with MIRACL module. Furthermore, there are **rdrand** instructions for the generation of randomic numbers. The functions involved in the MIRACL execution are also **sub_4022C0** and **sub_401A90**.



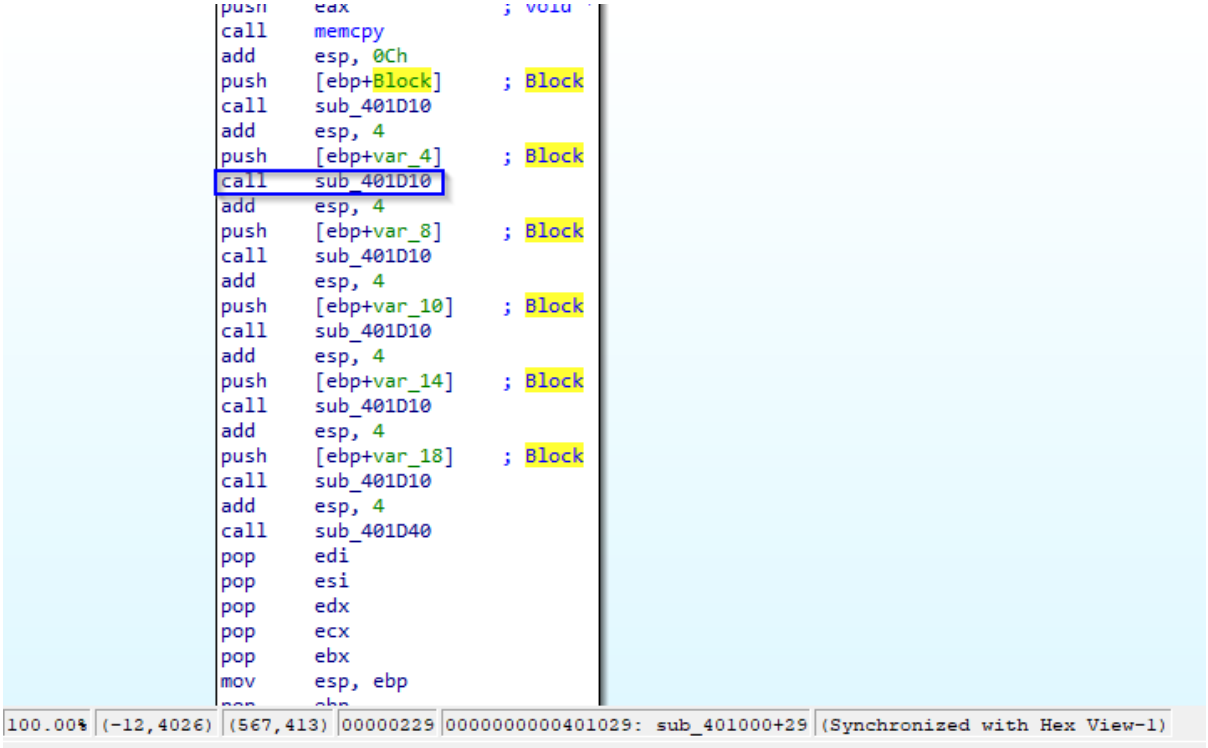


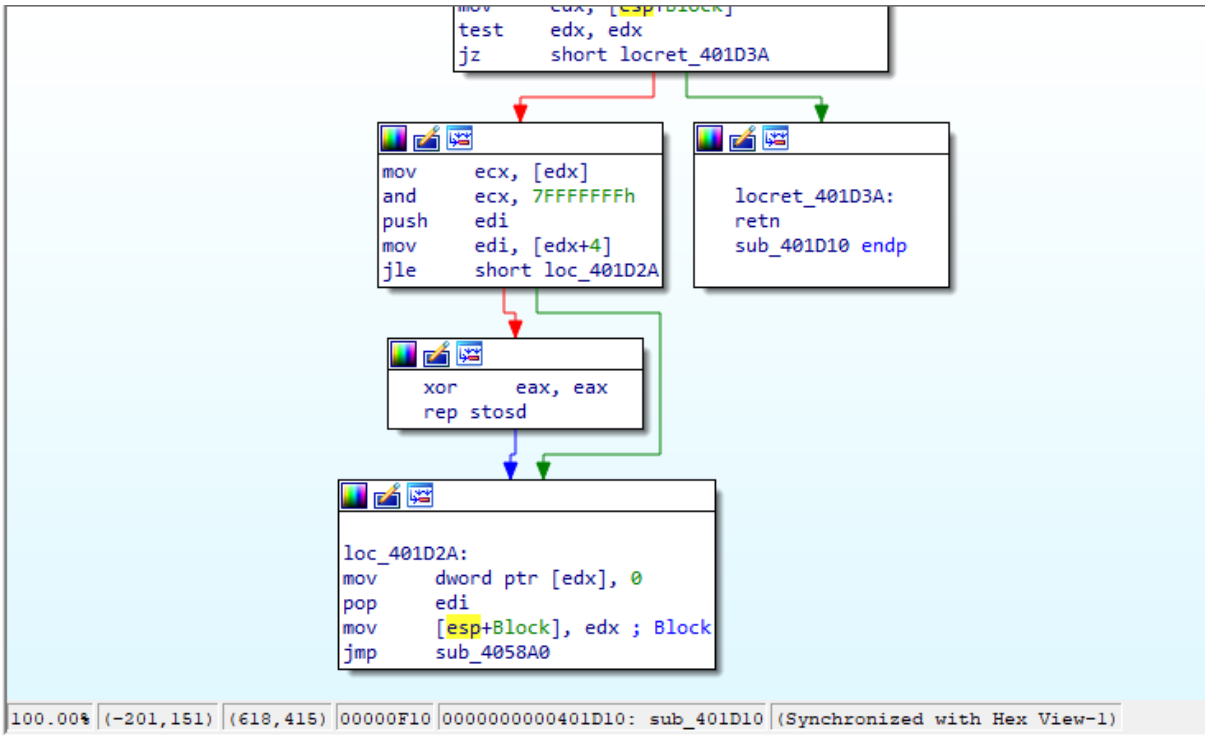

push edi
push 10h
push 400h
call sub_4022C0
add esp, 8
mov dword ptr [eax+234h], 10h
push 0
call sub_401A90
add esp, 4
mov [ebp+Block], eax
push 0
call sub_401A90
add esp, 4
mov [ebp+var_4], eax
push 0
call sub_401A90
add esp, 4
mov [ebp+var_8], eax
push 10001h
call sub_401A90
add esp, 4
mov [ebp+var_10], eax
push 0
call sub_401A90
add esp, 4
mov [ebp+var_14], eax
push 0
call sub_401A90
100.00% (-123,877) (811,285) 00000200 0000000000401000: sub_401000 (Synchronized with Hex View-1)


xor eax, eax
retn
loc_401AA2:
push esi
inc dword ptr [eax+1Ch]
mov eax, Block
mov ecx, [eax+1Ch]
cmp ecx, 18h
jge short loc_401AD4
mov dword ptr [eax+ecx*4+20h], 17h
mov eax, Block
mov ecx, [eax+240h]
test ecx, ecx
jz short loc_401AD4
call sub_401870
mov eax, Block
100.00% (193,206) (637,397) 00000C90 0000000000401A90: sub_401A90 (Synchronized with Hex View-1)

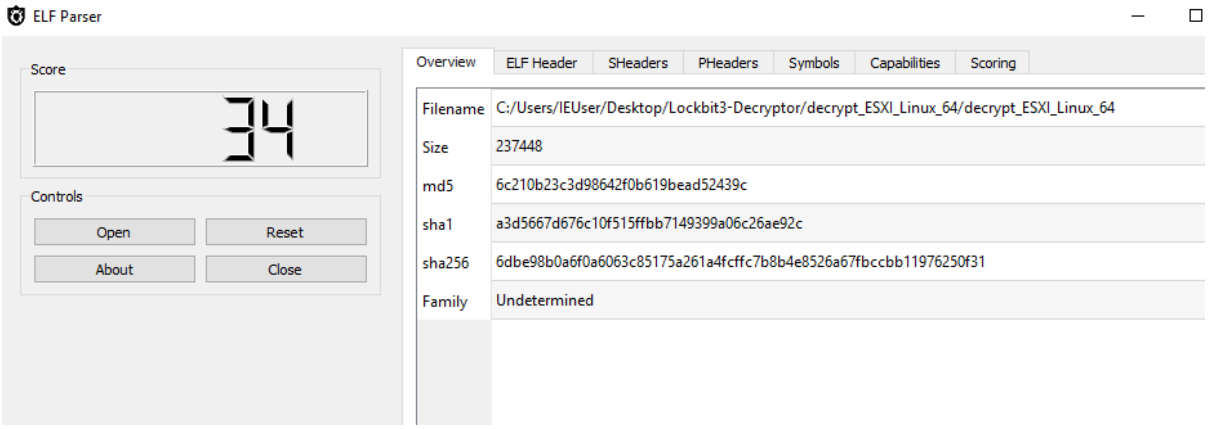


The function sub_401D10 is called recurrently and by having as attribute the variable associated to the public key. This function performs AND operations on the variable Block:





About the ELF file **decrypt_ESXI_Linux** is possible to verify that the given score by ELF Parser is **34**, so a quite "low" value which is an index that the file in question is heuristically known as little suspicious. There are evidences related to antidebugging (through ptrace) and some references that seem to be part of the phase of identification of the victim ID (also in this case there are details related to hashing functions).



Following some details extractable from the Read Only of the file, in which is possible to identify references to ***BLAKE2B_BLOCKBYTES, crypt_generichash_blake2b_final***, similarly to the Portable Executable version for Windows also in this case it is performed a call to hashing functionalities.

```
Read Only Segment (offset=0x28b40, size=43656, strings=37)
                String="""fD**~T"
                String=""fD"*~T*"
                String="%%oJ..r\"
                String="&jL&6Zl6?A~?"
                String="/dev/random"
                String="/dev/urandom"
                String="/proc/%d/cmdline"
                String="22Vd::Nt"
                String="2Vd2:Nt:"
                String=";d22Vt::N"
                String="=&&jL66Zl??A~"
                String="D""fT**~;"
                String="J%%o\..r8"
                String="L&&jl66Z~??A"
                String="LibsodiumDRGsodium/core.c"
                String="S->buflen <= BLAKE2B_BLOCKBYTES"
```

```
String="LibsodiumDRGsodium/core.c"
String="S->buflen <= BLAKE2B_BLOCKBYTES"
String="Vd22Nt::"
String="_sodium_malloc"
String="_unprotected_ptr_from_user_ptr(user_ptr) == unprotected_ptr"
String="blake2b_final"
String="crypto_generichash/blake2b/ref/blake2b-ref.c"
String="crypto_generichash/blake2b/ref/generichash_blake2b.c"
String="crypto_generichash_blake2b_final"
String="curve25519xsalsa20poly1305"
String="fD""~T**"
String="jL&&Zl66A~??"
String="locked == 0"
String="outlen <= (255)"
String="randombytes/sysrandom/randombytes_sysrandom.c"
String="safe_read"
String="size <= 9223372036854775807L"
```

Following the classification of various patterns identified in the ELF file, for example references to system and sysinfo(), randomic generation functions, manipulation of the processes (for example **fork()**) and antidebugging:

Category	Details
Shell	
	system() found
Random Functions	
	rand() found
	srand() found
Process Manipulation	
	daemon() found
	fork() found
	raise() found
Pipe Functions	
	pclose() found
	popen() found
Information Gathering	
	access() found
	sysinfo() found
File Functions	
	fclose() found
	unlink() found
Anti-Debug	
	ptrace detection found

Score	Reason
12	Process manipulation functions
2	Information gathering
10	Shell commands
10	Anti debug techniques

Here are the details of the magic number of the ELF analyzed, so **7f 45 4c 46** which identifies the typology of the ELF file:

Magic	7f 45 4c 46
Class	64-bit
Encoding	Little Endian
ELF Version	1
OS ABI	System V
ABI Version	0
Type	ET_EXEC
Machine	x86_64
Version	1
Entry Point	0x4023e0
PH Offset	64
SH Offset	235720
Flags	0x0
Header Size	64
PH Entry Size	56
PH Entries	9
SH Entry Size	64
SH Entries	27

String Index	26

Section Headers

Index	Name	Type	Flags	Virtual Address	Offset	Size	Link
0		K_NULL		0x0	0	0	0
1	.interp	K_PROGBITS	Alloc	0x400238	568	28	0
2	.note.ABI-tag	K_NOTE	Alloc	0x400254	596	32	0
3	.note.gnu.build-id	K_NOTE	Alloc	0x400274	628	36	0
4	.gnu.hash	K_GNU_HASH	Alloc	0x400298	664	64	5
5	.dynsym	K_DYNSYM	Alloc	0x4002d8	728	2688	6
6	.dynstr	K_STRTAB	Alloc	0x400d58	3416	1078	0
7	.gnu.version	K_VERSYM	Alloc	0x40118e	4494	224	5
8	.gnu.version_r	K_VERNEED	Alloc	0x401270	4720	128	6

Details

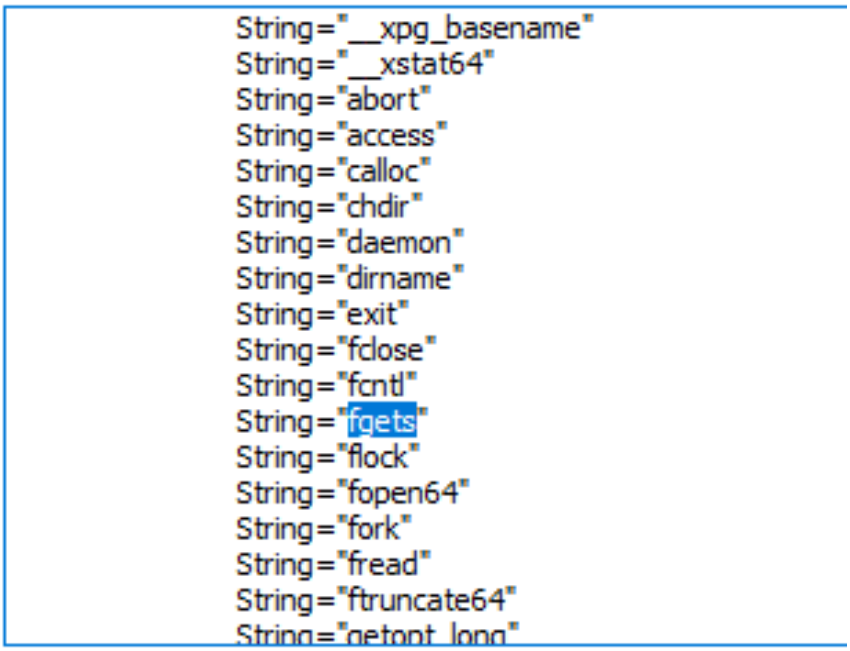
```
Interpreter (offset=0x238, size=28)
        Value="/lib64/ld-linux-x86-64.so.2"
```

6	.dynstr	K_STRTAB	Alloc	0x400d58	3416	1078	0
7	.gnu.version	K_VERSYM	Alloc	0x40118e	4494	224	5
8	.gnu.version_r	K_VERNEED	Alloc	0x401270	4720	128	6

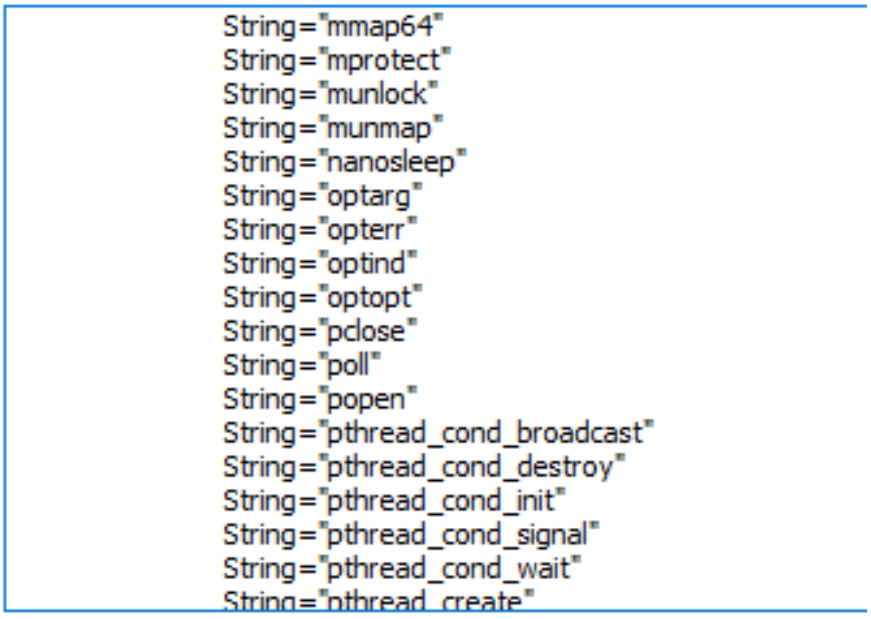
Details

```
String Table (offset=0xd58, size=1078, entries=108)
        String=""
        String="GLIBC_2.2.5"
        String="GLIBC_2.3"
        String="GLIBC_2.3.2"
        String="GLIBC_2.3.4"
        String="GLIBC_2.4"
        String="_Jv_RegisterClasses"
        String="__assert_fail"
        String="__ctype_tolower_loc"
        String="__errno_location"
        String="__fxstat"
        String="__fxstat64"
        String="__gmon_start__"
        String="__libc_start_main"
        String="__memcpy_chk"
        String="__stack_chk_fail"
```

Physiological, instead, is the presence of the call to a **fgets** function, so the ability to read the streams. In this specific case the possibility to read data and bytes to submit to the decryption phase:



Also in this case is possible to have the evidence of the creation of concurrential executions through threads (for example the functionality **pthread_cond_wait** can be used to put a block on a conditional variable):



Here are some references to randomic generations, for example **randombytes_sysrandom_implementation**

```
String="ptrace"
String="raise"
String="randombytes_sysrandom_implementation"
String="readlink"
String="rename"
String="setsid"
String="setvbuf"
String="snprintf"
String="sprintf"
String="srand"
String="stderr"
String="stdout"
String="strcasecmp"
String="strcat"
String="strchr"
String="strcmp"
String="strcpy"
String="strftime"
```

```
String="strftime"
String="strlen"
String="strncmp"
String="strncpy"
String="strrchr"
String="strstr"
String="strtok"
String="strtol"
String="sysconf"
String="sysinfo"
String="system"
String="size > (size_t) 0U"
String="sodium/utils.c"
String="sodium_crit_enter"
String="sysrandom"
String="x%oJ%.r\."
String="xxoJ%%r\..$8"
```

```
String Table (offset=0x397dd, size=229, entries=26)
String=""
String=".bss"
String=".comment"
String=".ctors"
String=".data"
String=".dtors"
String=".dynamic"
String=".dynstr"
String=".dynsym"
String=".eh_frame"
String=".eh_frame_hdr"
String=".fini"
String=".gnu.hash"
String=".gnu.version"
String=".gnu.version_r"
String=".got"
```

```
String=".init"
String=".interp"
String=".jcr"
String=".note.ABI-tag"
String=".note.gnu.build-id"
String=".rela.dyn"
String=".rela.plt"
String=".rodata"
String=".shstrtab"
String=".text"
```

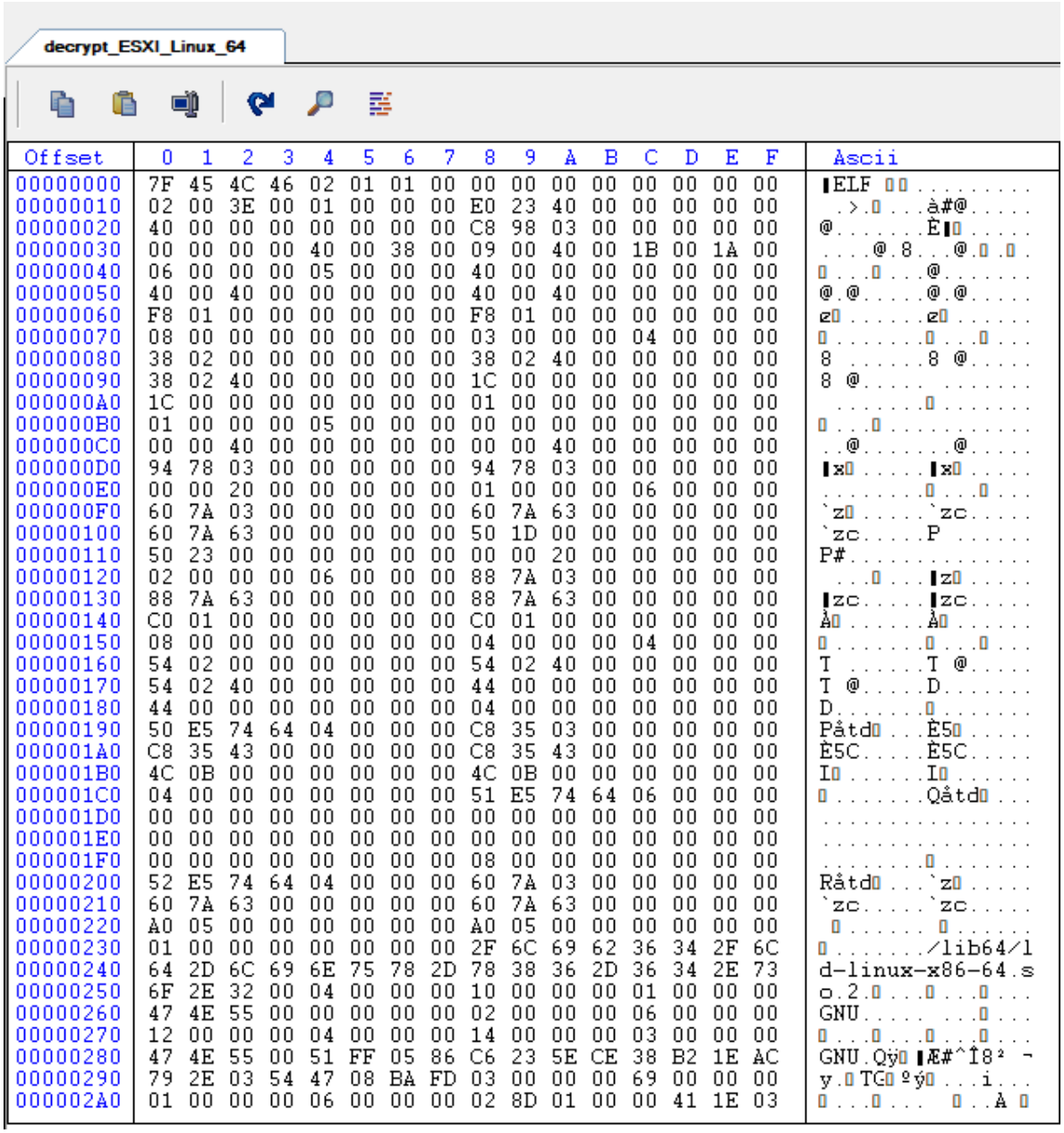
The call to a **chdir** function could permit to refer to different folders containing the files to manage:

	Type	Binding	Value
1	STT_NOTYPE	STB_LOCAL	0x0
2	STT_FUNC	STB_GLOBAL	daemon
3	STT_FUNC	STB_GLOBAL	mprotect
4	STT_FUNC	STB_GLOBAL	chdir
5	STT_FUNC	STB_GLOBAL	dirname
6	STT_FUNC	STB_GLOBAL	pthread_cond_destroy
7	STT_FUNC	STB_GLOBAL	memset
8	STT_FUNC	STB_GLOBAL	snprintf
9	STT_FUNC	STB_GLOBAL	setsid

	Type	Binding	Value
22	STT_FUNC	STB_GLOBAL	malloc
23	STT_FUNC	STB_GLOBAL	__libc_start_main
24	STT_FUNC	STB_GLOBAL	system
25	STT_FUNC	STB_GLOBAL	unlink
26	STT_FUNC	STB_GLOBAL	__memcpy_chk
27	STT_FUNC	STB_GLOBAL	gmtime
28	STT_FUNC	STB_GLOBAL	sysconf
29	STT_FUNC	STB_GLOBAL	pthread_mutex_init
30	STT_FUNC	STB_GLOBAL	fgets
31	STT_FUNC	STB_GLOBAL	__fxstat64
32	STT_FUNC	STB_GLOBAL	vfprintf
33	STT_FUNC	STB_GLOBAL	__strdup

73	STT_FUNC	STB_GLOBAL	srand
74	STT_FUNC	STB_GLOBAL	pthread_cond_wait
75	STT_FUNC	STB_GLOBAL	pthread_detach
76	STT_FUNC	STB_GLOBAL	__ctype_tolower_loc
77	STT_FUNC	STB_GLOBAL	memcmp
78	STT_FUNC	STB_GLOBAL	calloc
79	STT_FUNC	STB_GLOBAL	munmap
80	STT_FUNC	STB_GLOBAL	fclose
81	STT_FUNC	STB_GLOBAL	strncpy
82	STT_FUNC	STB_GLOBAL	__xstat64
83	STT_FUNC	STB_GLOBAL	lseek64
84	STT_FUNC	STB_GLOBAL	access

Following are the details of the hexadecimal of the ELF file in question, where is possible to identify the references to cryptography and ***Blake2B*** hash generation.



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	7F	45	4C	46	02	01	01	00	00	00	00	00	00	00	00	00	▌ELF ▯▯ .....
00000010	02	00	3E	00	01	00	00	00	E0	23	40	00	00	00	00	00	..>.▯ ...à#@.....
00000020	40	00	00	00	00	00	00	00	C8	98	03	00	00	00	00	00	@.....È▌▯ .....
00000030	00	00	00	00	40	00	38	00	09	00	40	00	1B	00	1A	00	....@.8...@.▯.▯.
00000040	06	00	00	00	05	00	00	00	40	00	00	00	00	00	00	00	▯...▯...@.....
00000050	40	00	40	00	00	00	00	00	40	00	40	00	00	00	00	00	@.@.....@.@.....
00000060	F8	01	00	00	00	00	00	00	F8	01	00	00	00	00	00	00	ø▯ ..... ø▯ .....
00000070	08	00	00	00	00	00	00	00	03	00	00	00	04	00	00	00	▯.....▯...▯...
00000080	38	02	00	00	00	00	00	00	38	02	40	00	00	00	00	00	8 .....8 @.....
00000090	38	02	40	00	00	00	00	00	1C	00	00	00	00	00	00	00	8 @..... .....
000000A0	1C	00	00	00	00	00	00	00	01	00	00	00	00	00	00	00	.....▯.....
000000B0	01	00	00	00	05	00	00	00	00	00	00	00	00	00	00	00	▯...▯.....
000000C0	00	00	40	00	00	00	00	00	00	00	40	00	00	00	00	00	..@.....@.....
000000D0	94	78	03	00	00	00	00	00	94	78	03	00	00	00	00	00	▌x▯ .....▌x▯ .....
000000E0	00	00	20	00	00	00	00	00	01	00	00	00	06	00	00	00	.....▯...▯...
000000F0	60	7A	03	00	00	00	00	00	60	7A	63	00	00	00	00	00	`z▯ .....`zc.....
00000100	60	7A	63	00	00	00	00	00	50	1D	00	00	00	00	00	00	`zc.....P .....
00000110	50	23	00	00	00	00	00	00	00	00	20	00	00	00	00	00	P#..... .....
00000120	02	00	00	00	06	00	00	00	88	7A	03	00	00	00	00	00	...▯...▌z▯ .....
00000130	88	7A	63	00	00	00	00	00	88	7A	63	00	00	00	00	00	▌zc.....▌zc.....
00000140	C0	01	00	00	00	00	00	00	C0	01	00	00	00	00	00	00	À▯ .....À▯ .....
00000150	08	00	00	00	00	00	00	00	04	00	00	00	04	00	00	00	▯.....▯...▯...
00000160	54	02	00	00	00	00	00	00	54	02	40	00	00	00	00	00	T .....T @.....
00000170	54	02	40	00	00	00	00	00	44	00	00	00	00	00	00	00	T @.....D.....
00000180	44	00	00	00	00	00	00	00	04	00	00	00	00	00	00	00	D.....▯.....
00000190	50	E5	74	64	04	00	00	00	C8	35	03	00	00	00	00	00	Påtd▯ ...È5▯ .....
000001A0	C8	35	43	00	00	00	00	00	C8	35	43	00	00	00	00	00	È5C.....È5C.....
000001B0	4C	0B	00	00	00	00	00	00	4C	0B	00	00	00	00	00	00	I▯ .....I▯ .....
000001C0	04	00	00	00	00	00	00	00	51	E5	74	64	06	00	00	00	▯.....Qåtd▯ ...
000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001F0	00	00	00	00	00	00	00	00	08	00	00	00	00	00	00	00	.....▯.....
00000200	52	E5	74	64	04	00	00	00	60	7A	03	00	00	00	00	00	Råtd▯ ...`z▯ .....
00000210	60	7A	63	00	00	00	00	00	60	7A	63	00	00	00	00	00	`zc.....`zc.....
00000220	A0	05	00	00	00	00	00	00	A0	05	00	00	00	00	00	00	▯..... ▯.....
00000230	01	00	00	00	00	00	00	00	2F	6C	69	62	36	34	2F	6C	▯...../lib64/l
00000240	64	2D	6C	69	6E	75	78	2D	78	38	36	2D	36	34	2E	73	d-linux-x86-64.s
00000250	6F	2E	32	00	04	00	00	00	10	00	00	00	01	00	00	00	o.2▯ ...▯ ...▯ ...
00000260	47	4E	55	00	00	00	00	00	02	00	00	00	06	00	00	00	GNU..... ...▯ ...
00000270	12	00	00	00	04	00	00	00	14	00	00	00	03	00	00	00	▯...▯...▯...▯...
00000280	47	4E	55	00	51	FF	05	86	C6	23	5E	CE	38	B2	1E	AC	GNU.Qÿ▯ ▌Æ#^Î8² ¬
00000290	79	2E	03	54	47	08	BA	FD	03	00	00	00	69	00	00	00	y.▯TG▯ºý▯ ...i...
000002A0	01	00	00	00	06	00	00	00	02	8D	01	00	00	41	1E	03	▯...▯... ▯..A ▯

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
0002AE90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0002AEA0	63	72	79	70	74	6F	5F	67	65	6E	65	72	69	63	68	61	crypto_genericha
0002AEB0	73	68	5F	62	6C	61	6B	65	32	62	5F	66	69	6E	61	6C	sh_blake2b_final
0002AEC0	00	4C	69	62	73	6F	64	69	75	6D	44	52	47	73	6F	64	.LibsodiumDRGsod
0002AED0	69	75	6D	2F	63	6F	72	65	2E	63	00	6C	6F	63	6B	65	ium/core.c.locke
0002AEE0	64	20	3D	3D	20	30	00	00	00	00	00	00	00	00	00	00	d.==.0..........
0002AEF0	73	6F	64	69	75	6D	5F	63	72	69	74	5F	65	6E	74	65	sodium_crit_ente
0002AF00	72	00	73	6F	64	69	75	6D	2F	75	74	69	6C	73	2E	63	r.sodium/utils.c
0002AF10	00	00	00	00	00	00	00	00	5F	75	6E	70	72	6F	74	65	........_unprote
0002AF20	63	74	65	64	5F	70	74	72	5F	66	72	6F	6D	5F	75	73	cted_ptr_from_us
0002AF30	65	72	5F	70	74	72	28	75	73	65	72	5F	70	74	72	29	er_ptr(user_ptr)
0002AF40	20	3D	3D	20	75	6E	70	72	6F	74	65	63	74	65	64	5F	.==.unprotected_
0002AF50	70	74	72	00	5F	73	6F	64	69	75	6D	5F	6D	61	6C	6C	ptr._sodium_mall
0002AF60	6F	63	00	73	79	73	72	61	6E	64	6F	6D	00	2F	64	65	oc.sysrandom./de
0002AF70	76	2F	72	61	6E	64	6F	6D	00	73	69	7A	65	20	3E	20	v/random.size.>.
0002AF80	28	73	69	7A	65	5F	74	29	20	30	55	00	73	69	7A	65	(size_t).0U.size
0002AF90	20	3C	3D	20	39	32	32	33	33	37	32	30	33	36	38	35	.<=.922337203685
0002AFA0	34	37	37	35	38	30	37	4C	00	2F	64	65	76	2F	75	72	4775807L./dev/ur
0002AFB0	61	6E	64	6F	6D	00	00	00	72	61	6E	64	6F	6D	62	79	andom...randomby
0002AFC0	74	65	73	2F	73	79	73	72	61	6E	64	6F	6D	2F	72	61	tes/sysrandom/ra
0002AFD0	6E	64	6F	6D	62	79	74	65	73	5F	73	79	73	72	61	6E	ndombytes_sysran
0002AFE0	64	6F	6D	2E	63	00	73	61	66	65	5F	72	65	61	64	00	dom.c.safe_read.
0002AFF0	63	75	72	76	65	32	35	35	31	39	78	73	61	6C	73	61	curve25519xsalsa
0002B000	32	30	70	6F	6C	79	31	33	30	35	00	00	00	00	00	00	20poly1305......
0002B010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0002B020	63	72	79	70	74	6F	5F	67	65	6E	65	72	69	63	68	61	crypto_genericha
0002B030	73	68	2F	62	6C	61	6B	65	32	62	2F	72	65	66	2F	62	sh/blake2b/ref/b
0002B040	6C	61	6B	65	32	62	2D	72	65	66	2E	63	00	00	00	00	lake2b-ref.c....
0002B050	53	2D	3E	62	75	66	6C	65	6E	20	3C	3D	20	42	4C	41	S->buflen.<=.BLA
0002B060	4B	45	32	42	5F	42	4C	4F	43	4B	42	59	54	45	53	00	KE2B_BLOCKBYTES.
0002B070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0002B080	62	6C	61	6B	65	32	62	5F	66	69	6E	61	6C	00	00	00	blake2b_final...
0002B090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0002B0A0	08	C9	BC	F3	67	E6	09	6A	3B	A7	CA	84	85	AE	67	BB	▯É¼ógæ.j;§Ê▮▮®g»
0002B0B0	2B	F8	94	FE	72	F3	6E	3C	F1	36	1D	5F	3A	F5	4F	A5	+ø▮þrón<ñ6 _:õO¥
0002B0C0	D1	82	E6	AD	7F	52	0E	51	1F	6C	3E	2B	8C	68	05	9B	Ñ▮æ-▮R▯Q l>+▮h▯▮
0002B0D0	6B	BD	41	FB	AB	D9	83	1F	79	21	7E	13	19	CD	E0	5B	k½Aû«Ù▮ y!~▯▯Íà[
0002B0E0	80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	▮...............
0002B0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0002B100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0002B110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0002B120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0002B130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

CONCLUSIONS:

The decryption tools analyzed perform a dynamic recognition of the cryptography context used on the compromised machine, in which the files were encrypted. It is present also an environment information gathering by the tool, in particular also of the user's settings NTUSER.dat. Some "cleaning" evidences related to some artifacts caused by the LockBit 3.0 infections, for example the restore of the Wallpaper and the icons, but also the remanipulation of some registry keys and some security identifiers. It is possible also to note that similarly to what happens with the encryption phase, one of the points that majorly characterizes the tool is the efficiency: specific threads are created for the execution of the decryption subroutine and to "point" to files obtained with an enumeration loop that will have to decrypt, by using the more efficient function SetFilePointerEx, instead of SetFilePointer.

Two fundamental evidences are securely related to the executions of bitswapping the files read in input by calling recurrently the function sub_401334, which includes also the XOR operations execution. The second evidence is instead related to the comparison between the value of eax register containing the MD5 hashing digest with hardcoded values in the decryptor with also the adding of AND and OR operations.

From the analysis of keygen.exe was possible to understand how, at the moment of the contextual creation of the ransomware and the decryptor, two keys are created: priv.key and pub.key.

The public key is used in the ransomware to encrypt the files of the victim. The private key, encoded in the decryptor, is used to decrypt the encrypted files only with the ransomware "linked" to the decryptor.

The decryptors are not "universal" but they are strictly related to the couple of public and private keys generated by keygen.exe

Technical Contributors:

Fabio Pensa

SoC Team Swascan

Contact Info

Milano
+39 0278620700
www.swascan.com
info@swascan.com
Via Fabio Filzi, 2b, 20063, Cernusco sul Naviglio, MI