McAfee Labs Threat Advisory

TDSS.rootkit

April 11, 2012

Summary

TDSS rootkit appeared around 2008 and is known for its ability to survive in the machine without being detected and the challenges it presents in terms of cleanup. There have been four versions of TDSS before this latest variant, and there have been improvements with every version in terms of being stealthy.

Detailed information about the rootkit, propagation vector, characteristics and mitigation etc are explained in the following sections.

Infection and Propagation Vectors

TDSS spreads by using affiliate marketing programs. Most affiliate marketing programs spreading malicious code use a Pay Per Install model which means the amount earned by the malware author depends on the number and the location of the machines it infects.

Characteristics and Symptoms

TDSS.e!rootkit

There are multiple variants of TDSS in the wild. All these variants exhibit different behavior. These are some of the behaviors exhibited by this variant *TDSS.e!rootkit*:

Upon execution of the dropper adjusts "SE_LOAD_DRIVER_PRIVILEGE", on success copies itself as a .DLL and calls **AddPrintProcessor,** which requests the system process "SPOOLSV.EXE" to load the specified library. It then creates a random service by executing ZwLoadDriver.

The malware hooks "KiDebugRoutine" which enables the malware to hide its traces in memory from a debugging program. When a debugging program tries to access the malware traces in memory through this hook, the malware intercepts the request and points to clean memory instead of the actual malicious code.

It then infects a windows component (.SYS file) which will enable it to start during system boot. The malware injects a thread in the kernel, so whenever the infected .SYS file is requested it always returns a clean one, instead of the one infected by the malware. In addition to this, the malware redirects searches. It also connects to its command and control server and sends information and receives commands. Connections to the following domains were observed on a infected machine:

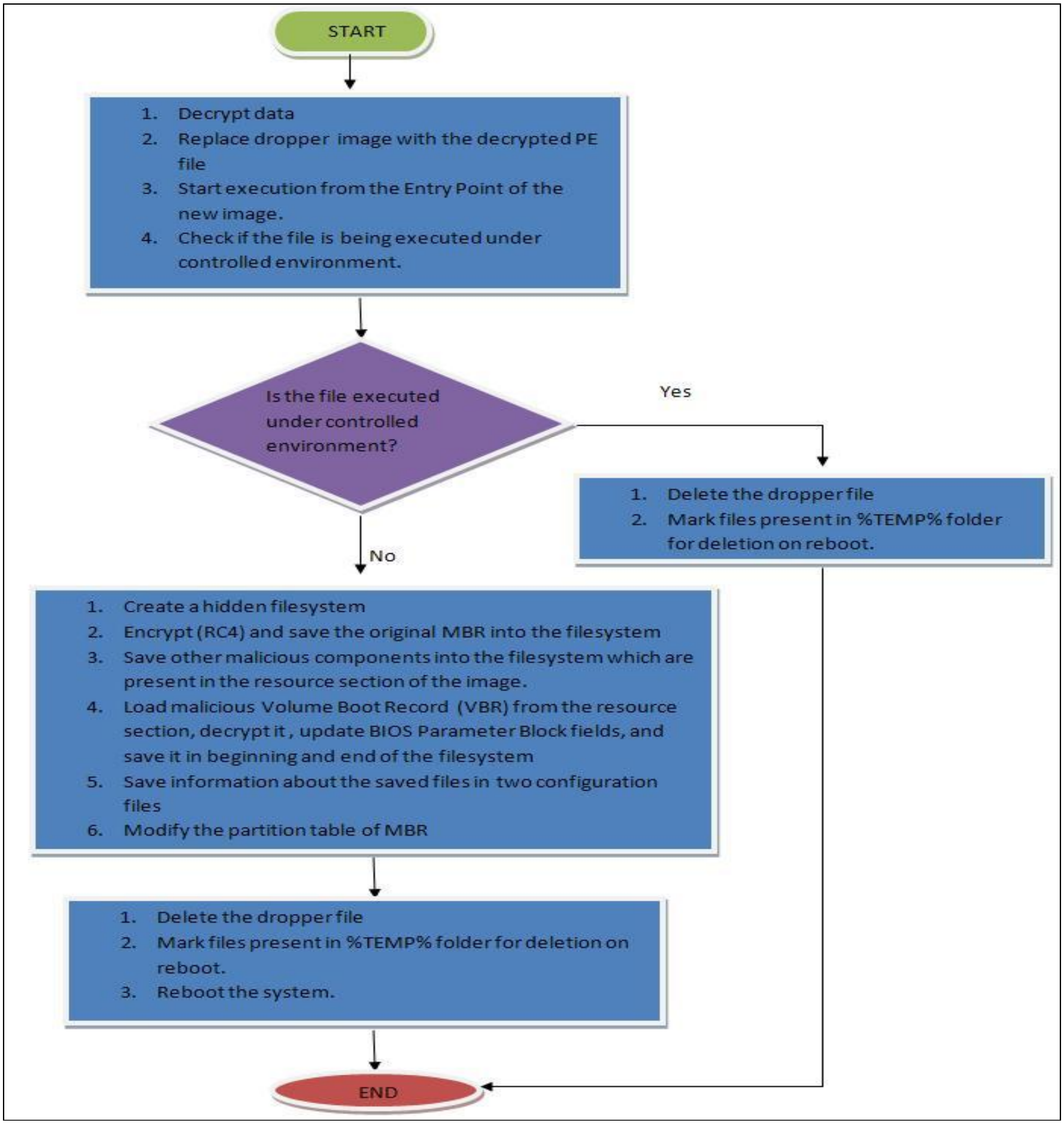
- https://nichtadden.in/
- https://91.212.226.67/
- https://li1i16b0.com/
- https://zz87jhfda88.com/
- https://n16fa53.com/
- https://01n02n4cx00.cc/
- https://lj1i16b0.com/
- http://clickpixelabn.com/
- http://thinksnotaeg.com/
- http://ijmgwarehouse.com/

- http://getbestbanner.com/
- http://pixelrotator.com/
- http://rf9akjgh716zzl.com/
- http://justgomediainc.in

TDSS.f!rootkit

Like other droppers of TDSS, TDSS.f dropper also carries actual infector in its resource section in an encrypted form. The actual infector is first decrypted and the dropper image is replaced with the decrypted infector.

Workflow Diagram for TDSS:



1. Decrypt data
2. Replace dropper image with the decrypted PE file
3. Start execution from the Entry Point of the new image.
4. Check if the file is being executed under controlled environment.

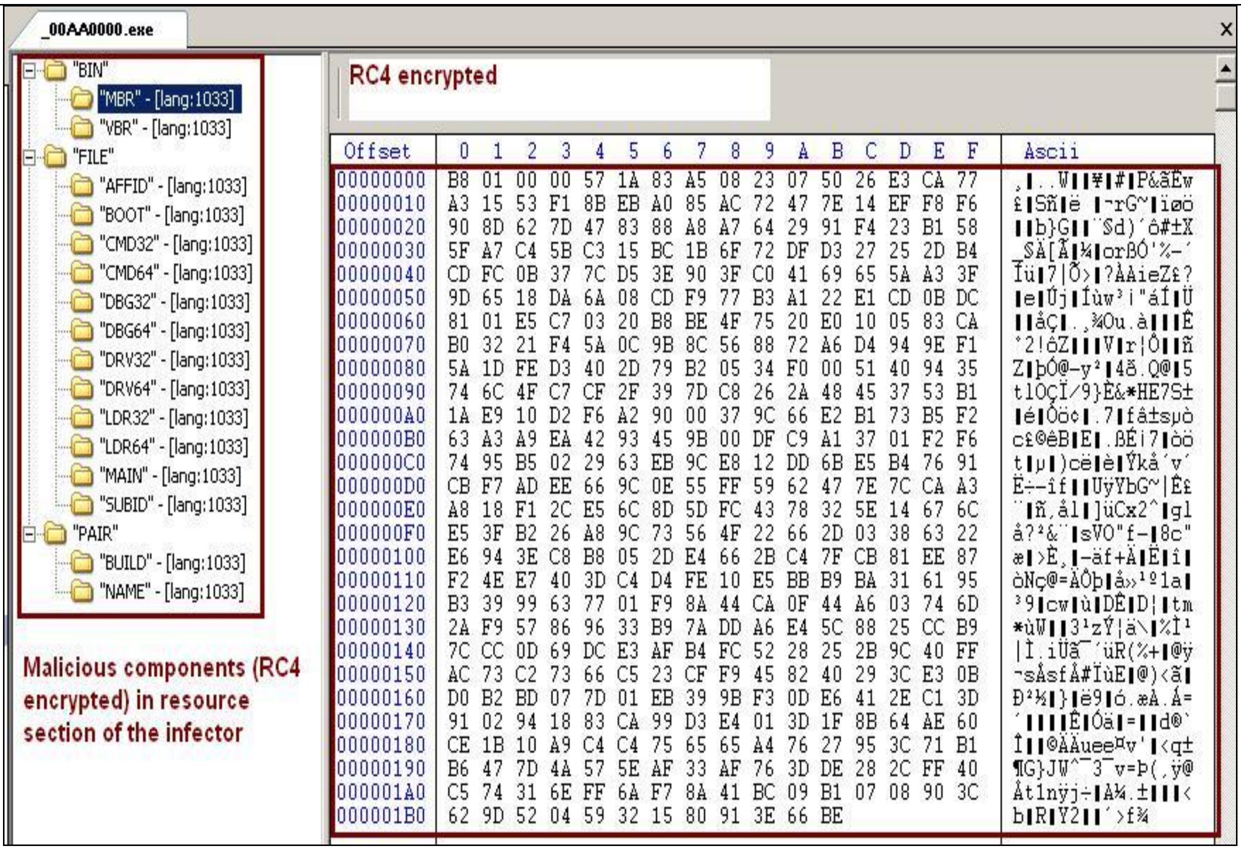
1. Delete the dropper file
2. Mark files present in %TEMP% folder for deletion on reboot.

1. Create a hidden filesystem
2. Encrypt (RC4) and save the original MBR into the filesystem
3. Save other malicious components into the filesystem which are present in the resource section of the image.
4. Load malicious Volume Boot Record (VBR) from the resource section, decrypt it , update BIOS Parameter Block fields, and save it in beginning and end of the filesystem
5. Save information about the saved files in two configuration files
6. Modify the partition table of MBR

1. Delete the dropper file
2. Mark files present in %TEMP% folder for deletion on reboot.
3. Reboot the system.

(1.0 TDSS Workflow)

Remaining malicious components are present in infector’s resource section as shown below:



_00AA0000.exe

```
"BIN"
    "MBR" - [lang:1033]
    "VBR" - [lang:1033]
"FILE"
    "AFFID" - [lang:1033]
    "BOOT" - [lang:1033]
    "CMD32" - [lang:1033]
    "CMD64" - [lang:1033]
    "DBG32" - [lang:1033]
    "DBG64" - [lang:1033]
    "DRV32" - [lang:1033]
    "DRV64" - [lang:1033]
    "LDR32" - [lang:1033]
    "LDR64" - [lang:1033]
    "MAIN" - [lang:1033]
    "SUBID" - [lang:1033]
"PAIR"
    "BUILD" - [lang:1033]
    "NAME" - [lang:1033]
```

Malicious components (RC4 encrypted) in resource section of the infector

RC4 encrypted

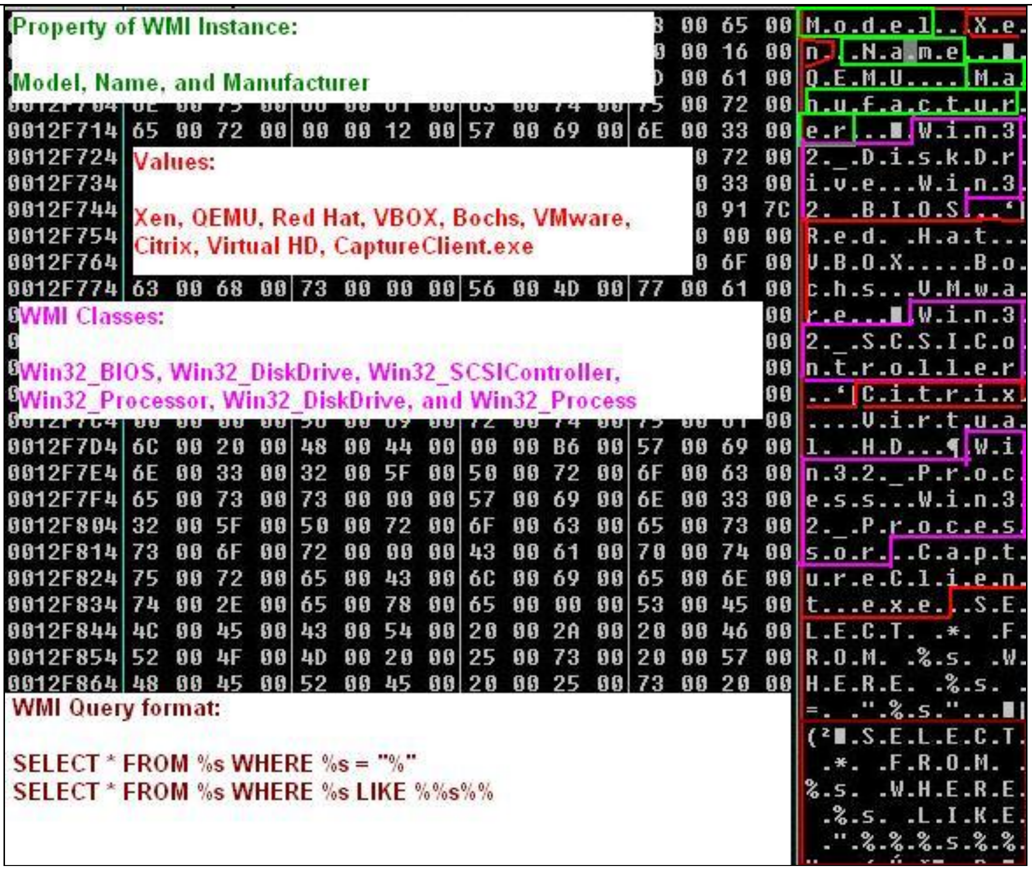
```
Offset    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   Ascii
00000000  B8 01 00 00 57 1A 83 A5 08 23 07 50 26 E3 CA 77  ¸▮..W▮▮¥▮#▮P&ãÊw
00000010  A3 15 53 F1 8B EB A0 85 AC 72 47 7E 14 EF F8 F6  £▮Sñ▮ë ▮¬rG~▮ïøö
00000020  90 8D 62 7D 47 83 88 A8 A7 64 29 91 F4 23 B1 58  ▮▮b}G▮▮¨§d)´ô#±X
00000030  5F A7 C4 5B C3 15 BC 1B 6F 72 DF D3 27 25 2D B4  _§Ä[Ã▮¼▮orßÓ'%-´
00000040  CD FC 0B 37 7C D5 3E 90 3F C0 41 69 65 5A A3 3F  Íü▮7|Õ>▮?ÀAieZ£?
00000050  9D 65 18 DA 6A 08 CD F9 77 B3 A1 22 E1 CD 0B DC  ▮e▮Új▮Íùw³¡"áÍ▮Ü
00000060  81 01 E5 C7 03 20 B8 BE 4F 75 20 E0 10 05 83 CA  ▮▮åÇ▮.¸¾Ou.à▮▮▮Ê
00000070  B0 32 21 F4 5A 0C 9B 8C 56 88 72 A6 D4 94 9E F1  °2!ôZ▮▮▮V▮r¦Ô▮▮ñ
00000080  5A 1D FE D3 40 2D 79 B2 05 34 F0 00 51 40 94 35  Z▮þÓ@-y²▮4ð.Q@▮5
00000090  74 6C 4F C7 CF 2F 39 7D C8 26 2A 48 45 37 53 B1  tlOÇÏ/9}È&*HE7S±
000000A0  1A E9 10 D2 F6 A2 90 00 37 9C 66 E2 B1 73 B5 F2  ▮é▮Òö¢▮.7▮fâ±sµò
000000B0  63 A3 A9 EA 42 93 45 9B 00 DF C9 A1 37 01 F2 F6  c£©êB▮E▮.ßÉ¡7▮òö
000000C0  74 95 B5 02 29 63 EB 9C E8 12 DD 6B E5 B4 76 91  t▮µ▮)cë▮è▮Ýkå´v´
000000D0  CB F7 AD EE 66 9C 0E 55 FF 59 62 47 7E 7C CA A3  Ë÷-îf▮▮UÿYbG~|Ê£
000000E0  A8 18 F1 2C E5 6C 8D 5D FC 43 78 32 5E 14 67 6C  ¨▮ñ,ål▮]üCx2^▮gl
000000F0  E5 3F B2 26 A8 9C 73 56 4F 22 66 2D 03 38 63 22  å?²&¨▮sVO"f-▮8c"
00000100  E6 94 3E C8 B8 05 2D E4 66 2B C4 7F CB 81 EE 87  æ▮>È¸▮-äf+Ä▮Ë▮î▮
00000110  F2 4E E7 40 3D C4 D4 FE 10 E5 BB B9 BA 31 61 95  òNç@=ÄÔþ▮å»¹º1a▮
00000120  B3 39 99 63 77 01 F9 8A 44 CA 0F 44 A6 03 74 6D  ³9▮cw▮ù▮DÊ▮D¦▮tm
00000130  2A F9 57 86 96 33 B9 7A DD A6 E4 5C 88 25 CC B9  *ùW▮▮3¹zÝ¦ä\▮%Ì¹
00000140  7C CC 0D 69 DC E3 AF B4 FC 52 28 25 2B 9C 40 FF  |Ì.iÜã¯´üR(%+▮@ÿ
00000150  AC 73 C2 73 66 C5 23 CF F9 45 82 40 29 3C E3 0B  ¬sÂsfÅ#ÏùE▮@)<ã▮
00000160  D0 B2 BD 07 7D 01 EB 39 9B F3 0D E6 41 2E C1 3D  Ð²½▮}▮ë9▮ó.æA.Á=
00000170  91 02 94 18 83 CA 99 D3 E4 01 3D 1F 8B 64 AE 60  ´▮▮▮▮Ê▮Óä▮=▮▮d®`
00000180  CE 1B 10 A9 C4 C4 75 65 65 A4 76 27 95 3C 71 B1  Î▮▮©ÄÄuee¤v'▮<q±
00000190  B6 47 7D 4A 57 5E AF 33 AF 76 3D DE 28 2C FF 40  ¶G}JW^¯3¯v=Þ(,ÿ@
000001A0  C5 74 31 6E FF 6A F7 8A 41 BC 09 B1 07 08 90 3C  Åt1nÿj÷▮A¼.±▮▮▮<
000001B0  62 9D 52 04 59 32 15 80 91 3E 66 BE              b▮R▮Y2▮▮´>f¾
```

(1.1 Resource Section of infector)

These resources are loaded whenever required. Initially, it loads few resources like “BUILD”, “NAME” of “PAIR” type and “SUBID”, “MAIN” of type “FILE”.

Some variants of TDSS.f before infection checks if it is running in controlled environment.

It connects to the “Root\Cimv2” WMI interface, retrieve system resource information like instances of Win32_BIOS, Win32_DiskDrive, Win32_SCSIController, Win32_Processor, Win32_Process and check if the malware is being executed in controlled environment.



Property of WMI Instance:

Model, Name, and Manufacturer

Values:

Xen, QEMU, Red Hat, VBOX, Bochs, VMware, Citrix, Virtual HD, CaptureClient.exe

WMI Classes:

Win32_BIOS, Win32_DiskDrive, Win32_SCSIController, Win32_Processor, Win32_DiskDrive, and Win32_Process

WMI Query format:

SELECT * FROM %s WHERE %s = "%"
SELECT * FROM %s WHERE %s LIKE %%s%%

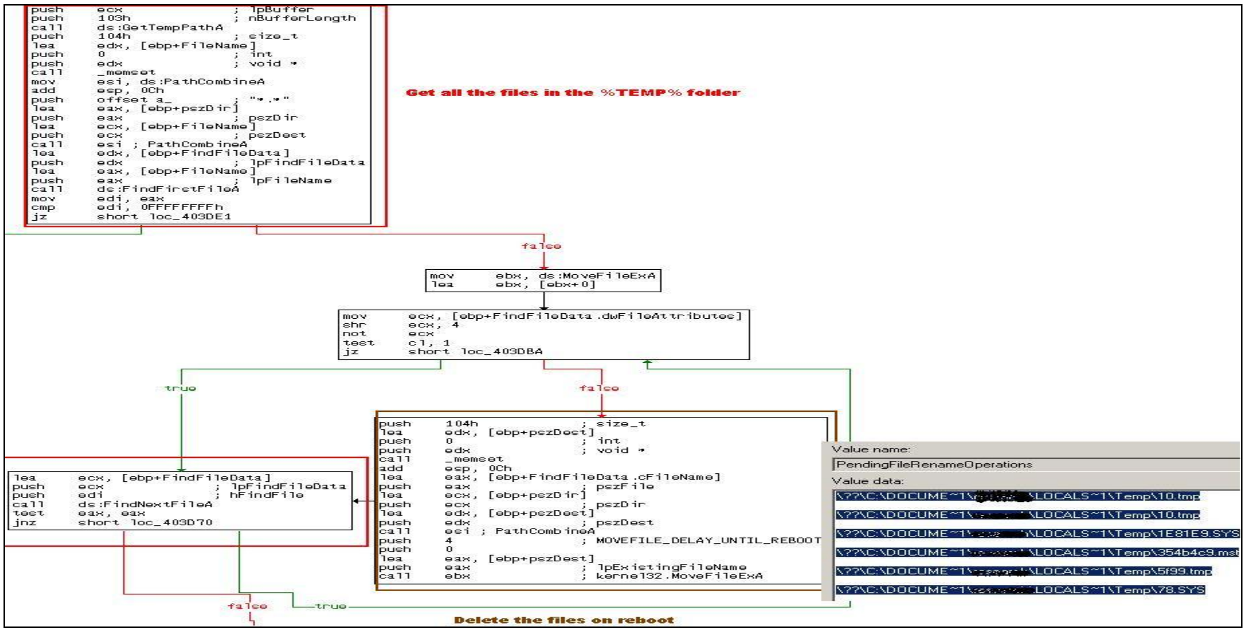
```
                                                  00 65 00 M.o.d.e.l...X.e.
                                                  00 16 00 n...N.a.m.e.....
                                                  00 61 00 Q.E.M.U.....M.a.
                                                  00 72 00 n.u.f.a.c.t.u.r.
0012F714 65 00 72 00|00 00 12 00|57 00 69 00|6E 00 33 00 e.r.....W.i.n.3.
0012F724                                          0 72 00 2._.D.i.s.k.D.r.
0012F734                                          0 33 00 i.v.e...W.i.n.3.
0012F744                                          0 91 7C 2. .B.I.O.S...‘|
0012F754                                          0 00 00 R.e.d. .H.a.t...
0012F764                                          0 6F 00 V.B.O.X.....B.o.
0012F774 63 00 68 00|73 00 00 00|56 00 4D 00|77 00 61 00 c.h.s...V.M.w.a.
                                                     00 r.e.....W.i.n.3.
                                                     00 2._.S.C.S.I.C.o.
                                                     00 n.t.r.o.l.l.e.r.
                                                     00 ..‘.C.i.t.r.i.x.
                                                     00 ....V.i.r.t.u.a.
0012F7D4 6C 00 20 00|48 00 44 00|00 00 B6 00|57 00 69 00 l. .H.D...¶.W.i.
0012F7E4 6E 00 33 00|32 00 5F 00|50 00 72 00|6F 00 63 00 n.3.2._.P.r.o.c.
0012F7F4 65 00 73 00|73 00 00 00|57 00 69 00|6E 00 33 00 e.s.s...W.i.n.3.
0012F804 32 00 5F 00|50 00 72 00|6F 00 63 00|65 00 73 00 2._.P.r.o.c.e.s.
0012F814 73 00 6F 00|72 00 00 00|43 00 61 00|70 00 74 00 s.o.r...C.a.p.t.
0012F824 75 00 72 00|65 00 43 00|6C 00 69 00|65 00 6E 00 u.r.e.C.l.i.e.n.
0012F834 74 00 2E 00|65 00 78 00|65 00 00 00|53 00 45 00 t...e.x.e...S.E.
0012F844 4C 00 45 00|43 00 54 00|20 00 2A 00|20 00 46 00 L.E.C.T. .*. .F.
0012F854 52 00 4F 00|4D 00 20 00|25 00 73 00|20 00 57 00 R.O.M. .%.s. .W.
0012F864 48 00 45 00|52 00 45 00|20 00 25 00|73 00 20 00 H.E.R.E. .%.s. .
                                                          =. .".%.s."...▮|
                                                          (²▮.S.E.L.E.C.T.
                                                           .*. .F.R.O.M. .
                                                          %.s. .W.H.E.R.E.
                                                           .%.s. .L.I.K.E.
                                                           .".%.%.%.s.%.%.
```

(1.2 WMI Instance)

If the malware discovers that it is running in controlled environment,

- It skips the execution path which is responsible for MBR partition table modification and creation of hidden filesystem, wherein it keeps original MBR and other malicious components in encrypted form.

- It directly calls a routine which removes its traces by deleting the dropper and other files from the %TEMP% folder as shown below:



Get all the files in the %TEMP% folder

```
push    ecx                     ; lpBuffer
push    103h                    ; nBufferLength
call    ds:GetTempPathA
push    104h                    ; size_t
lea     edx, [ebp+FileName]
push    0                       ; int
push    edx                     ; void *
call    _memset
mov     esi, ds:PathCombineA
add     esp, 0Ch
push    offset a_               ; "*.*"
lea     eax, [ebp+pszDir]
push    eax                     ; pszDir
lea     ecx, [ebp+FileName]
push    ecx                     ; pszDest
call    esi ; PathCombineA
lea     edx, [ebp+FindFileData]
push    edx                     ; lpFindFileData
lea     eax, [ebp+FileName]
push    eax                     ; lpFileName
call    ds:FindFirstFileA
mov     edi, eax
cmp     edi, 0FFFFFFFFh
jz      short loc_403DE1
```

```
mov     ebx, ds:MoveFileExA
lea     ebx, [ebx+0]
```

```
mov     ecx, [ebp+FindFileData.dwFileAttributes]
shr     ecx, 4
not     ecx
test    cl, 1
jz      short loc_403DBA
```

```
lea     ecx, [ebp+FindFileData]
push    ecx                     ; lpFindFileData
push    edi                     ; hFindFile
call    ds:FindNextFileA
test    eax, eax
jnz     short loc_403D70
```

```
push    104h                    ; size_t
lea     edx, [ebp+pszDest]
push    0                       ; int
push    edx                     ; void *
call    _memset
add     esp, 0Ch
lea     eax, [ebp+FindFileData.cFileName]
push    eax                     ; pszFile
lea     ecx, [ebp+pszDir]
push    ecx                     ; pszDir
lea     edx, [ebp+pszDest]
push    edx                     ; pszDest
call    esi ; PathCombineA
push    4                       ; MOVEFILE_DELAY_UNTIL_REBOOT
push    0
lea     eax, [ebp+pszDest]
push    eax                     ; lpExistingFileName
call    ebx                     ; kernel32.MoveFileExA
```

Value name:
PendingFileRenameOperations
Value data:

```
\??\C:\DOCUME~1\[redacted]\LOCALS~1\Temp\10.tmp
\??\C:\DOCUME~1\[redacted]\LOCALS~1\Temp\10.tmp
\??\C:\DOCUME~1\[redacted]\LOCALS~1\Temp\1E81E9.SYS
\??\C:\DOCUME~1\[redacted]\LOCALS~1\Temp\354b4c9.msi
\??\C:\DOCUME~1\[redacted]\LOCALS~1\Temp\5f99.tmp
\??\C:\DOCUME~1\[redacted]\LOCALS~1\Temp\78.SYS
```

Delete the files on reboot

(1.3 code flow 1)

When the malware is executed on a physical machine, it retrieves handle of the first HDD **“\\.\PhysicalDrive0**” and uses **IOCTL_SCSI_PASS_THROUGH_DIRECT I/O** control code to read and write to the HDD.

For data transfer operations, a buffer with alignment matching the adapter device is required.

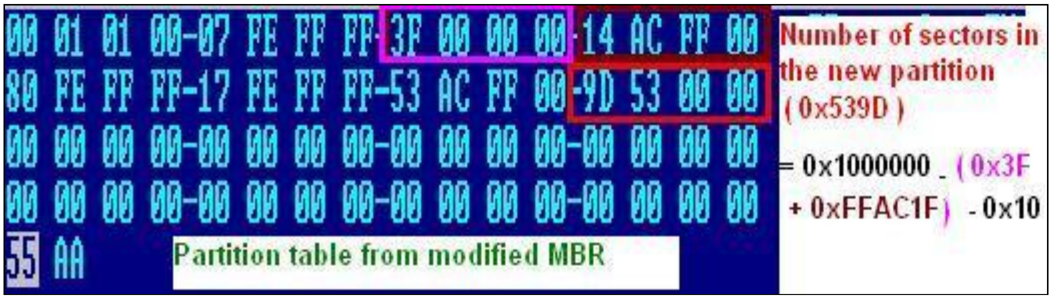
Therefore it first retrieves AlignmentMask using the **IOCTL_STORAGE_QUERY_PROPERTY** control code and then retrieves the capacity of the device using **IOCTL_SCSI_PASS_THROUGH_DIRECT** control code.

Infection Flow
It first reads MBR into memory, parses the partition table and look for the bootable partition.

After identifying the bootable partition, it computes absolute number of sectors by adding the LBA of the first absolute sector of active partition and number of sectors in the partition.

The summation of the above two is then subtracted from 0x1000000. It then calculates the number of sectors to be used in the filesystem to be created.

Number of sectors = 0x1000000 – (LBA of first absolute sector of the active partition + Number of sectors in the partition) – 0x10



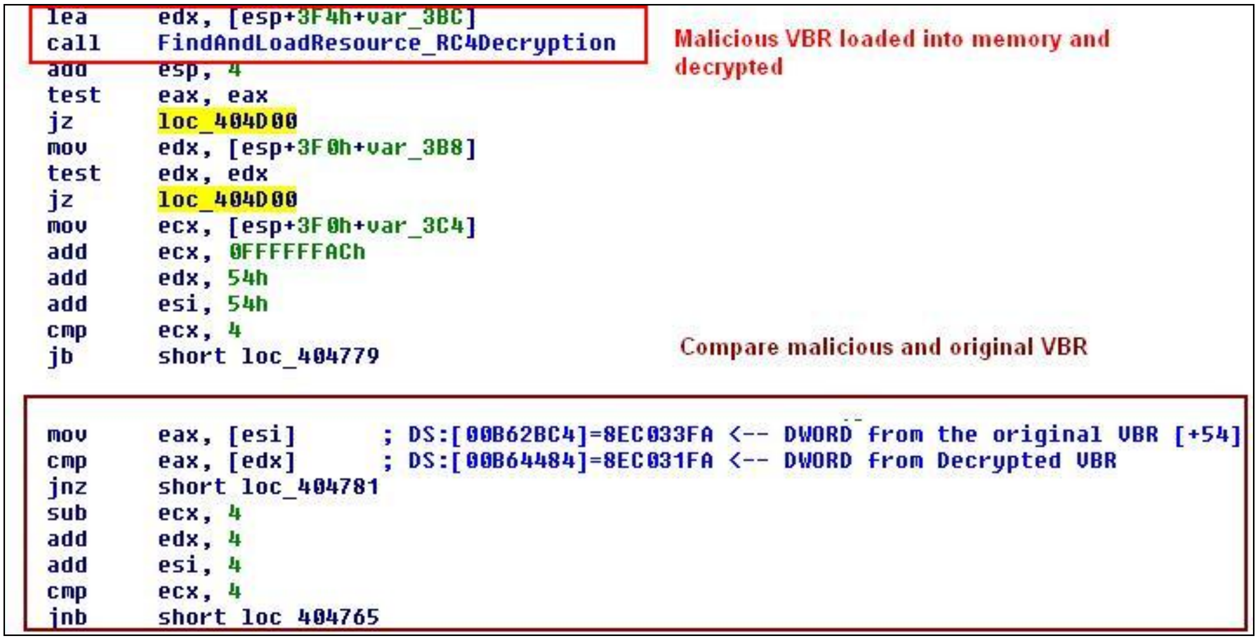
```
00 01 01 00-07 FE FF FF-3F 00 00 00-14 AC FF 00
80 FE FF FF-17 FE FF FF-53 AC FF 00-9D 53 00 00
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
55 AA
```

Number of sectors in the new partition (0x539D) = 0x1000000 - (0x3F + 0xFFAC1F) - 0x10

Partition table from modified MBR

(1.4 Modified MBR)

It then loads, and decrypts the resource named “**vbr**” of type “**BIN**” in memory. To avoid re-infection, it compares the malicious VBR code with the original VBR code as shown below:



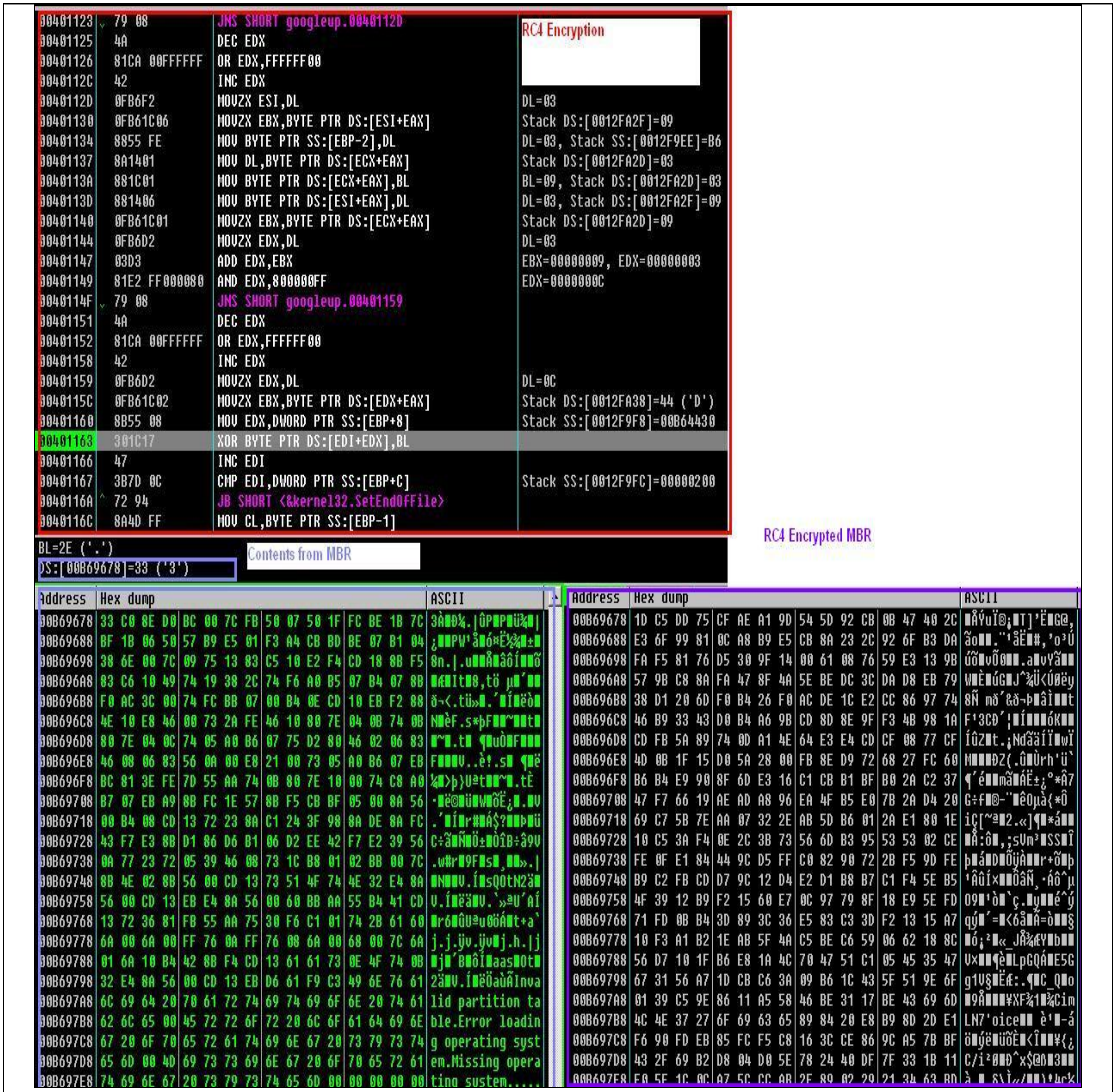
```
lea     edx, [esp+3F4h+var_3BC]
call    FindAndLoadResource_RC4Decryption
add     esp, 4
test    eax, eax
jz      loc_404D00
mov     edx, [esp+3F0h+var_3B8]
test    edx, edx
jz      loc_404D00
mov     ecx, [esp+3F0h+var_3C4]
add     ecx, 0FFFFFFACh
add     edx, 54h
add     esi, 54h
cmp     ecx, 4
jb      short loc_404779

mov     eax, [esi]      ; DS:[00B62BC4]=8EC033FA <-- DWORD from the original VBR [+54]
cmp     eax, [edx]      ; DS:[00B64484]=8EC031FA <-- DWORD from Decrypted VBR
jnz     short loc_404781
sub     ecx, 4
add     edx, 4
add     esi, 4
cmp     ecx, 4
jnb     short loc_404765
```

Malicious VBR loaded into memory and decrypted

Compare malicious and original VBR

(1.5 Code flow 2)

The original MBR which was read previously into memory is encrypted, written into the hidden file system and the storage information is saved in secondary configuration as shown below:

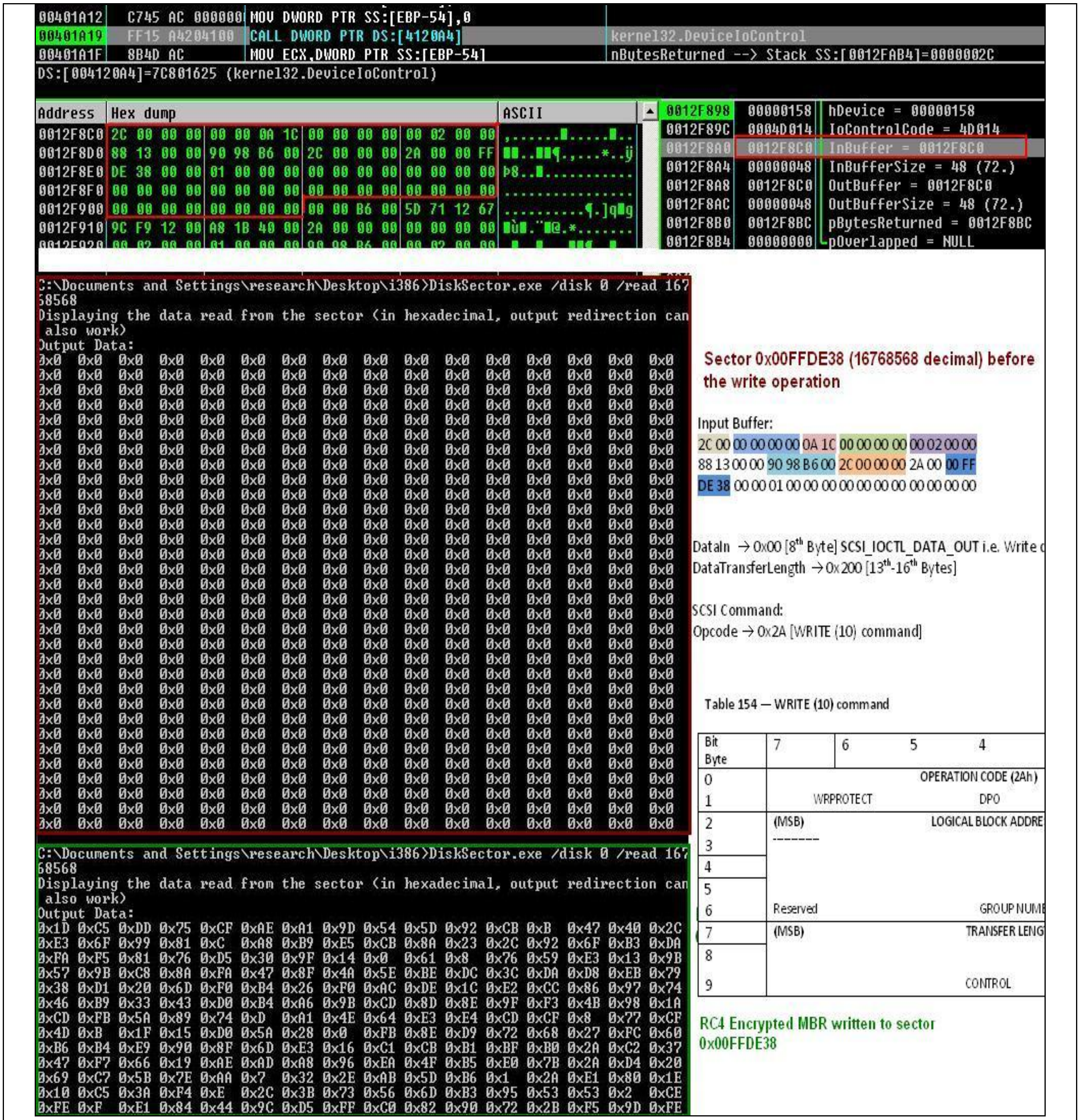


```
00401123 v 79 08           JNS SHORT googleup.0040112D
00401125   4A              DEC EDX
00401126   81CA 00FFFFFF   OR EDX,FFFFFF00
0040112C   42              INC EDX
0040112D   0FB6F2          MOVZX ESI,DL                         DL=03
00401130   0FB61C06        MOVZX EBX,BYTE PTR DS:[ESI+EAX]      Stack DS:[0012FA2F]=09
00401134   8855 FE         MOV BYTE PTR SS:[EBP-2],DL           DL=03, Stack SS:[0012F9EE]=B6
00401137   8A1401          MOV DL,BYTE PTR DS:[ECX+EAX]         Stack DS:[0012FA2D]=03
0040113A   881C01          MOV BYTE PTR DS:[ECX+EAX],BL         BL=09, Stack DS:[0012FA2D]=03
0040113D   881406          MOV BYTE PTR DS:[ESI+EAX],DL         DL=03, Stack DS:[0012FA2F]=09
00401140   0FB61C01        MOVZX EBX,BYTE PTR DS:[ECX+EAX]      Stack DS:[0012FA2D]=09
00401144   0FB6D2          MOVZX EDX,DL                         DL=03
00401147   03D3            ADD EDX,EBX                          EBX=00000009, EDX=00000003
00401149   81E2 FF000080   AND EDX,800000FF                     EDX=0000000C
0040114F v 79 08           JNS SHORT googleup.00401159
00401151   4A              DEC EDX
00401152   81CA 00FFFFFF   OR EDX,FFFFFF00
00401158   42              INC EDX
00401159   0FB6D2          MOVZX EDX,DL                         DL=0C
0040115C   0FB61C02        MOVZX EBX,BYTE PTR DS:[EDX+EAX]      Stack DS:[0012FA38]=44 ('D')
00401160   8B55 08         MOV EDX,DWORD PTR SS:[EBP+8]         Stack SS:[0012F9F8]=00B64430
00401163   301C17          XOR BYTE PTR DS:[EDI+EDX],BL
00401166   47              INC EDI
00401167   3B7D 0C         CMP EDI,DWORD PTR SS:[EBP+C]         Stack SS:[0012F9FC]=00000200
0040116A ^ 72 94           JB SHORT <&kernel32.SetEndOfFile>
0040116C   8A4D FF         MOV CL,BYTE PTR SS:[EBP-1]
BL=2E ('.')
DS:[00B69678]=33 ('3')
```

```
Address  Hex dump                                         ASCII
00B69678 33 C0 8E D0 BC 00 7C FB 50 07 50 1F FC BE 1B 7C 3À■Ð¼.|ûP■P■ü¾■|
00B69688 BF 1B 06 50 57 B9 E5 01 F3 A4 CB BD BE 07 B1 04 ¿■■PW¹å■ó¤Ë½¾■±■
00B69698 38 6E 00 7C 09 75 13 83 C5 10 E2 F4 CD 18 8B F5 8n.|.u■■Å■âôÍ■■õ
00B696A8 83 C6 10 49 74 19 38 2C 74 F6 A0 B5 07 B4 07 8B ■Æ■It■8,tö µ■´■■
00B696B8 F0 AC 3C 00 74 FC BB 07 00 B4 0E CD 10 EB F2 88 ð¬<.tü»■.´■Í■ëò■
00B696C8 4E 10 E8 46 00 73 2A FE 46 10 80 7E 04 0B 74 0B N■èF.s*þF■■~■■t■
00B696D8 80 7E 04 0C 74 05 A0 B6 07 75 D2 80 46 02 06 83 ■~■.t■ ¶■uÒ■F■■■
00B696E8 46 08 06 83 56 0A 00 E8 21 00 73 05 A0 B6 07 EB F■■■V..è!.s■ ¶■ë
00B696F8 BC 81 3E FE 7D 55 AA 74 0B 80 7E 10 00 74 C8 A0 ¼■>þ}Uªt■■~■.tÈ
00B69708 B7 07 EB A9 8B FC 1E 57 8B F5 CB BF 05 00 8A 56 ·■ë©■ü■W■õË¿■.■V
00B69718 00 B4 08 CD 13 72 23 8A C1 24 3F 98 8A DE 8A FC .´■Í■r#■Á$?■■Þ■ü
00B69728 43 F7 E3 8B D1 86 D6 B1 06 D2 EE 42 F7 E2 39 56 C÷ã■Ñ■Ö±■ÒîB÷â9V
00B69738 0A 77 23 72 05 39 46 08 73 1C B8 01 02 BB 00 7C .w#r■9F■s■¸■■».|
00B69748 8B 4E 02 8B 56 00 CD 13 73 51 4F 74 4E 32 E4 8A ■N■■V.Í■sQOtN2ä■
00B69758 56 00 CD 13 EB E4 8A 56 00 60 BB AA 55 B4 41 CD V.Í■ëä■V.`»ªU´AÍ
00B69768 13 72 36 81 FB 55 AA 75 30 F6 C1 01 74 2B 61 60 ■r6■ûUªu0öÁ■t+a`
00B69778 6A 00 6A 00 FF 76 0A FF 76 08 6A 00 68 00 7C 6A j.j.ÿv.ÿv■j.h.|j
00B69788 01 6A 10 B4 42 8B F4 CD 13 61 61 73 0E 4F 74 0B ■j■´B■ôÍ■aas■Ot■
00B69798 32 E4 8A 56 00 CD 13 EB D6 61 F9 C3 49 6E 76 61 2ä■V.Í■ëÖaùÃInva
00B697A8 6C 69 64 20 70 61 72 74 69 74 69 6F 6E 20 74 61 lid partition ta
00B697B8 62 6C 65 00 45 72 72 6F 72 20 6C 6F 61 64 69 6E ble.Error loadin
00B697C8 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 g operating syst
00B697D8 65 6D 00 4D 69 73 73 69 6E 67 20 6F 70 65 72 61 em.Missing opera
00B697E8 74 69 6E 67 20 73 79 73 74 65 6D 00 00 00 00 00 ting system.....
```

```
Address  Hex dump                                         ASCII
00B69678 1D C5 DD 75 CF AE A1 9D 54 5D 92 CB 0B 47 40 2C ■ÅÝuÏ®¡■T]’Ë■G@,
00B69688 E3 6F 99 81 0C A8 B9 E5 CB 8A 23 2C 92 6F B3 DA ão■■.¨¹åË■#,’o³Ú
00B69698 FA F5 81 76 D5 30 9F 14 00 61 08 76 59 E3 13 9B úõ■vÕ0■■.a■vYã■■
00B696A8 57 9B C8 8A FA 47 8F 4A 5E BE DC 3C DA D8 EB 79 W■È■úG■J^¾Ü<ÚØëy
00B696B8 38 D1 20 6D F0 B4 26 F0 AC DE 1C E2 CC 86 97 74 8Ñ mð´&ð¬Þ■âÌ■■t
00B696C8 46 B9 33 43 D0 B4 A6 9B CD 8D 8E 9F F3 4B 98 1A F¹3CÐ´¦■Í■■■óK■■
00B696D8 CD FB 5A 89 74 0D A1 4E 64 E3 E4 CD CF 08 77 CF ÍûZ■t.¡NdãäÍÏ■wÏ
00B696E8 4D 0B 1F 15 D0 5A 28 00 FB 8E D9 72 68 27 FC 60 M■■■ÐZ(.û■Ùrh'ü`
00B696F8 B6 B4 E9 90 8F 6D E3 16 C1 CB B1 BF B0 2A C2 37 ¶´é■■mã■ÁË±¿°*Â7
00B69708 47 F7 66 19 AE AD A8 96 EA 4F B5 E0 7B 2A D4 20 G÷f■®-¨■êOµà{*Ô
00B69718 69 C7 5B 7E AA 07 32 2E AB 5D B6 01 2A E1 80 1E iÇ[~ª■2.«]¶■*á■■
00B69728 10 C5 3A F4 0E 2C 3B 73 56 6D B3 95 53 53 02 CE ■Å:ô■,;sVm³■SS■Î
00B69738 FE 0F E1 84 44 9C D5 FF C0 82 90 72 2B F5 9D FE þ■á■D■ÕÿÀ■■r+õ■þ
00B69748 B9 C2 FB CD D7 9C 12 D4 E2 D1 B8 B7 C1 F4 5E B5 ¹ÂûÍ×■■ÔâÑ¸·Áô^µ
00B69758 4F 39 12 B9 F2 15 60 E7 0C 97 79 8F 18 E9 5E FD O9■¹ò■`ç.■y■■é^ý
00B69768 71 FD 0B B4 3D 89 3C 36 E5 83 C3 3D F2 13 15 A7 qý■´=■<6å■Ã=ò■■§
00B69778 10 F3 A1 B2 1E AB 5F 4A C5 BE C6 59 06 62 18 8C ■ó¡²■«_JÅ¾ÆY■b■■
00B69788 56 D7 10 1F B6 E8 1A 4C 70 47 51 C1 05 45 35 47 V×■■¶è■LpGQÁ■E5G
00B69798 67 31 56 A7 1D CB C6 3A 09 B6 1C 43 5F 51 9E 6F g1V§■ËÆ:.¶■C_Q■o
00B697A8 01 39 C5 9E 86 11 A5 58 46 BE 31 17 BE 43 69 6D ■9Å■■■¥XF¾1■¾Cim
00B697B8 4C 4E 37 27 6F 69 63 65 89 84 20 E8 B9 8D 2D E1 LN7'oice■■ è¹■-á
00B697C8 F6 90 FD EB 85 FC F5 C8 16 3C CE 86 9C A5 7B BF ö■ýë■üõÈ■<Î■■¥{¿
00B697D8 43 2F 69 B2 D8 04 D0 5E 78 24 40 DF 7F 33 1B 11 C/i²Ø■Ð^x$@ß■3■■
00B697E8 E0 5F 1C 0C A7 5C CC AB 2F 89 02 29 21 34 63 BD à_■■§\Ì«/■■)!4c½
```

(1.6 Original MBR is encrypted before it is saved into hidden file system)



```
00401A12   C745 AC 000000 MOV DWORD PTR SS:[EBP-54],0
00401A19   FF15 A4204100  CALL DWORD PTR DS:[4120A4]     kernel32.DeviceIoControl
00401A1F   8B4D AC        MOV ECX,DWORD PTR SS:[EBP-54]  nBytesReturned --> Stack SS:[0012FAB4]=0000002C
DS:[004120A4]=7C801625 (kernel32.DeviceIoControl)
```

```
Address  Hex dump                                         ASCII
0012F8C0 2C 00 00 00 00 00 0A 1C 00 00 00 00 00 02 00 00  ,.......█.....█..
0012F8D0 88 13 00 00 90 98 B6 00 2C 00 00 00 2A 00 00 FF  ██..██¶.,...*..ÿ
0012F8E0 DE 38 00 00 01 00 00 00 00 00 00 00 00 00 00 00  Þ8..█...........
0012F8F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0012F900 00 00 00 00 00 00 00 00 00 00 B6 00 5D 71 12 67  ..........¶.]q█g
0012F910 9C F9 12 00 A8 1B 40 00 2A 00 00 00 00 00 00 00  █ù█.¨█@.*.......
0012F920 00 02 00 00 01 00 00 00 00 98 B6 00 00 02 00 00
```

```
0012F898  00000158  hDevice = 00000158
0012F89C  0004D014  IoControlCode = 4D014
0012F8A0  0012F8C0  InBuffer = 0012F8C0
0012F8A4  00000048  InBufferSize = 48 (72.)
0012F8A8  0012F8C0  OutBuffer = 0012F8C0
0012F8AC  00000048  OutBufferSize = 48 (72.)
0012F8B0  0012F8BC  pBytesReturned = 0012F8BC
0012F8B4  00000000  pOverlapped = NULL
```

```
C:\Documents and Settings\research\Desktop\i386>DiskSector.exe /disk 0 /read 167
68568
Displaying the data read from the sector (in hexadecimal, output redirection can
 also work)
Output Data:
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0  0x0
```

Sector 0x00FFDE38 (16768568 decimal) before the write operation

Input Buffer:
2C 00 00 00 00 00 0A 1C 00 00 00 00 00 02 00 00
88 13 00 00 90 98 B6 00 2C 00 00 00 2A 00 00 FF
DE 38 00 00 01 00 00 00 00 00 00 00 00 00 00 00

DataIn → 0x00 [8th Byte] SCSI_IOCTL_DATA_OUT i.e. Write c
DataTransferLength → 0x200 [13th-16th Bytes]

SCSI Command:
Opcode → 0x2A [WRITE (10) command]

Table 154 — WRITE (10) command

Bit Byte	7	6	5	4
0				OPERATION CODE (2Ah)
1		WRPROTECT		DPO
2	(MSB)			LOGICAL BLOCK ADDRE
3	-------			
4				
5				
6	Reserved			GROUP NUMB
7	(MSB)			TRANSFER LENG
8				
9				CONTROL

```
C:\Documents and Settings\research\Desktop\i386>DiskSector.exe /disk 0 /read 167
68568
Displaying the data read from the sector (in hexadecimal, output redirection can
 also work)
Output Data:
0x1D 0xC5 0xDD 0x75 0xCF 0xAE 0xA1 0x9D 0x54 0x5D 0x92 0xCB 0xB  0x47 0x40 0x2C
0xE3 0x6F 0x99 0x81 0xC  0xA8 0xB9 0xE5 0xCB 0x8A 0x23 0x2C 0x92 0x6F 0xB3 0xDA
0xFA 0xF5 0x81 0x76 0xD5 0x30 0x9F 0x14 0x0  0x61 0x8  0x76 0x59 0xE3 0x13 0x9B
0x57 0x9B 0xC8 0x8A 0xFA 0x47 0x8F 0x4A 0x5E 0xBE 0xDC 0x3C 0xDA 0xD8 0xEB 0x79
0x38 0xD1 0x20 0x6D 0xF0 0xB4 0x26 0xF0 0xAC 0xDE 0x1C 0xE2 0xCC 0x86 0x97 0x74
0x46 0xB9 0x33 0x43 0xD0 0xB4 0xA6 0x9B 0xCD 0x8D 0x8E 0x9F 0xF3 0x4B 0x98 0x1A
0xCD 0xFB 0x5A 0x89 0x74 0xD  0xA1 0x4E 0x64 0xE3 0xE4 0xCD 0xCF 0x8  0x77 0xCF
0x4D 0xB  0x1F 0x15 0xD0 0x5A 0x28 0x0  0xFB 0x8E 0xD9 0x72 0x68 0x27 0xFC 0x60
0xB6 0xB4 0xE9 0x90 0x8F 0x6D 0xE3 0x16 0xC1 0xCB 0xB1 0xBF 0xB0 0x2A 0xC2 0x37
0x47 0xF7 0x66 0x19 0xAE 0xAD 0xA8 0x96 0xEA 0x4F 0xB5 0xE0 0x7B 0x2A 0xD4 0x20
0x69 0xC7 0x5B 0x7E 0xAA 0x7  0x32 0x2E 0xAB 0x5D 0xB6 0x1  0x2A 0xE1 0x80 0x1E
0x10 0xC5 0x3A 0xF4 0xE  0x2C 0x3B 0x73 0x56 0x6D 0xB3 0x95 0x53 0x53 0x2  0xCE
0xFE 0xF  0xE1 0x84 0x44 0x9C 0xD5 0xFF 0xC0 0x82 0x90 0x72 0x2B 0xF5 0x9D 0xFE
```

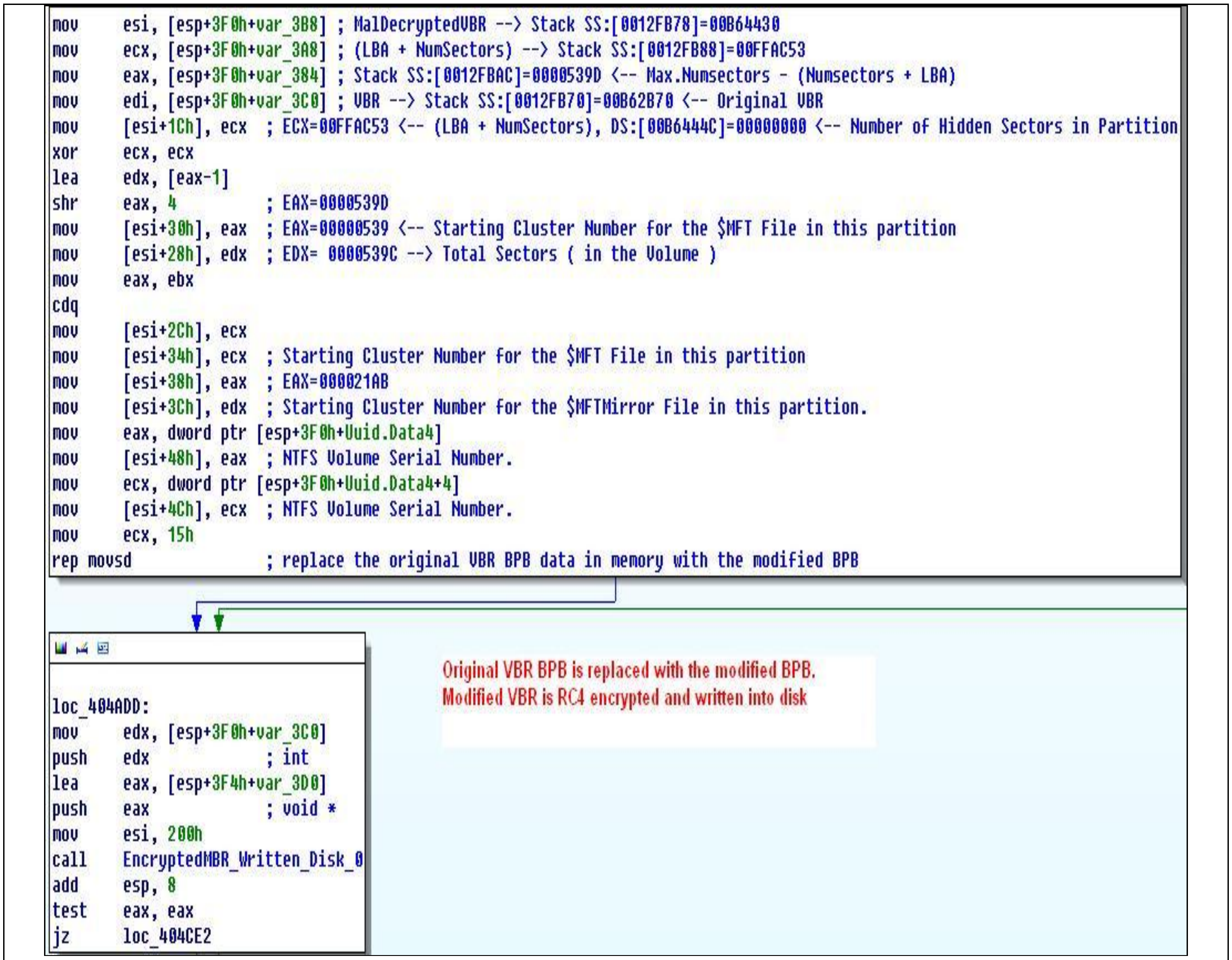
RC4 Encrypted MBR written to sector 0x00FFDE38

(1.7 Sector where the original MBR is saved in encrypted form)

The BIOS Parameter Block (BPB) of the boot sector is then updated with information like:

- Number of Hidden Sectors in Partition [offset: 0x1C]
- Sector Number of the File System Information Sector [offset: 0x30]
- Total Sectors (in the Volume) [offset: 0x28]. This value is 1 sector less than the total number of sectors in the volume's partition table entry, because an NTFS "Backup Sector" is not considered part of the NTFS Volume.
- Starting Cluster Number for the $MFTMirror File in this partition [offset: 0x38].
  - This field is populated with the data which is used to identify the sector which contains primary configuration file.
- NTFS Volume Serial Number [offset: 0x48]
  - This field is populated with the decryption key.

After BPB modification, VBR is encrypted, written into the hidden file system and the storage information are saved in the secondary configuration file as shown below:

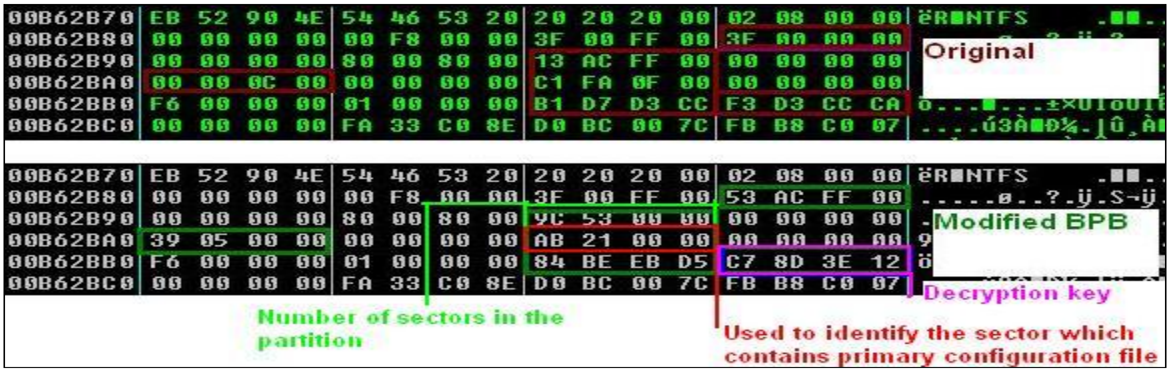


```
mov     esi, [esp+3F0h+var_3B8] ; MalDecryptedVBR --> Stack SS:[0012FB78]=00B64430
mov     ecx, [esp+3F0h+var_3A8] ; (LBA + NumSectors) --> Stack SS:[0012FB88]=00FFAC53
mov     eax, [esp+3F0h+var_384] ; Stack SS:[0012FBAC]=0000539D <-- Max.Numsectors - (Numsectors + LBA)
mov     edi, [esp+3F0h+var_3C0] ; VBR --> Stack SS:[0012FB70]=00B62B70 <-- Original VBR
mov     [esi+1Ch], ecx  ; ECX=00FFAC53 <-- (LBA + NumSectors), DS:[00B6444C]=00000000 <-- Number of Hidden Sectors in Partition
xor     ecx, ecx
lea     edx, [eax-1]
shr     eax, 4          ; EAX=0000539D
mov     [esi+30h], eax  ; EAX=00000539 <-- Starting Cluster Number for the $MFT File in this partition
mov     [esi+28h], edx  ; EDX= 0000539C --> Total Sectors ( in the Volume )
mov     eax, ebx
cdq
mov     [esi+2Ch], ecx
mov     [esi+34h], ecx  ; Starting Cluster Number for the $MFT File in this partition
mov     [esi+38h], eax  ; EAX=000021AB
mov     [esi+3Ch], edx  ; Starting Cluster Number for the $MFTMirror File in this partition.
mov     eax, dword ptr [esp+3F0h+Uuid.Data4]
mov     [esi+48h], eax  ; NTFS Volume Serial Number.
mov     ecx, dword ptr [esp+3F0h+Uuid.Data4+4]
mov     [esi+4Ch], ecx  ; NTFS Volume Serial Number.
mov     ecx, 15h
rep movsd               ; replace the original VBR BPB data in memory with the modified BPB
```

```
loc_404ADD:
mov     edx, [esp+3F0h+var_3C0]
push    edx             ; int
lea     eax, [esp+3F4h+var_3D0]
push    eax             ; void *
mov     esi, 200h
call    EncryptedMBR_Written_Disk_0
add     esp, 8
test    eax, eax
jz      loc_404CE2
```

Original VBR BPB is replaced with the modified BPB.
Modified VBR is RC4 encrypted and written into disk

(1.8 Boot Sector BIOS Parameter Block updated)



```
00B62B70 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 ëR■NTFS    .■■..
00B62B80 00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00
00B62B90 00 00 00 00 80 00 80 00 13 AC FF 00 00 00 00 00
00B62BA0 00 00 0C 00 00 00 00 00 C1 FA 0F 00 00 00 00 00
00B62BB0 F6 00 00 00 01 00 00 00 B1 D7 D3 CC F3 D3 CC CA ö...■...±×ÓÌóÓÌÊ
00B62BC0 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB B8 C0 07 ....ú3À■Ð¼.|û¸À■
```

Original

```
00B62B70 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 ëR■NTFS    .■■..
00B62B80 00 00 00 00 00 F8 00 00 3F 00 FF 00 53 AC FF 00 .....ø..?.ÿ.S¬ÿ.
00B62B90 00 00 00 00 80 00 80 00 9C 53 00 00 00 00 00 00 .
00B62BA0 39 05 00 00 00 00 00 00 AB 21 00 00 00 00 00 00 9
00B62BB0 F6 00 00 00 01 00 00 00 84 BE EB D5 C7 8D 3E 12 ö
00B62BC0 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB B8 C0 07
```

Modified BPB

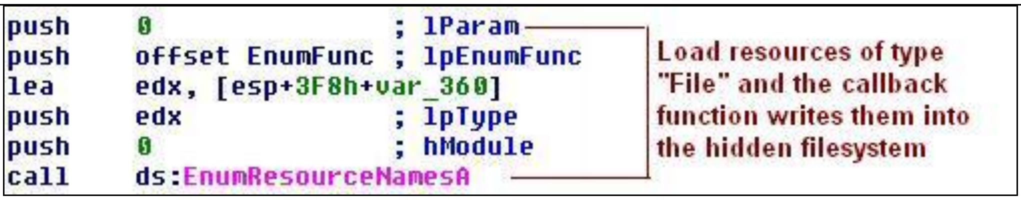
Decryption key

Number of sectors in the partition

Used to identify the sector which contains primary configuration file

(1.9 code flow 3)

Now, all the resources of type “FILE” are searched in the module, transfers control to a callback function which writes them into the hidden filesystem and update information about each file or data which is saved into the hidden file system into the secondary configuration file as shown below:

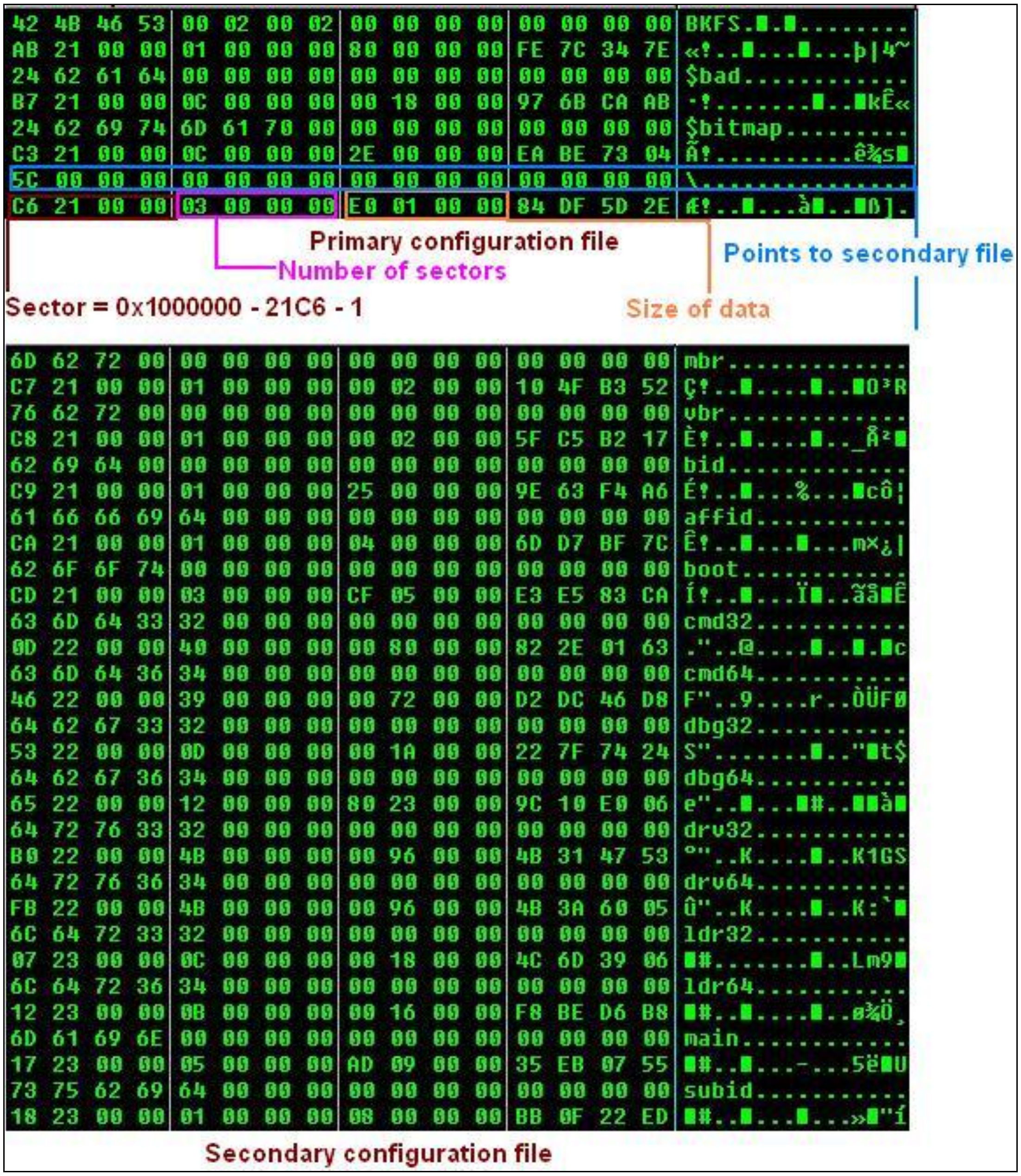


```
push    0                   ; lParam
push    offset EnumFunc     ; lpEnumFunc
lea     edx, [esp+3F8h+var_360]
push    edx                 ; lpType
push    0                   ; hModule
call    ds:EnumResourceNamesA
```

Load resources of type "File" and the callback function writes them into the hidden filesystem

(1.10 code flow 4)

The configuration file consists of a set of blocks. Each block is 0x20 bytes long. Primary configuration file has four blocks whereas the secondary file has fifteen blocks. Each block begins with 0x10 bytes long name which indicates type of data which is saved in the disk. It is followed by 4 bytes long element which helps in identifying the sector wherein data has been saved. This is followed by "Number of Sectors" and "Data Size" elements as shown below:



```
42 4B 46 53|00 02 00 02|00 00 00 00|00 00 00 00|BKFS.■.■........
AB 21 00 00|01 00 00 00|80 00 00 00|FE 7C 34 7E|«!..■...■...þ|4~
24 62 61 64|00 00 00 00|00 00 00 00|00 00 00 00|$bad............
B7 21 00 00|0C 00 00 00|00 18 00 00|97 6B CA AB|·!.......■..■kÊ«
24 62 69 74|6D 61 70 00|00 00 00 00|00 00 00 00|$bitmap.........
C3 21 00 00|0C 00 00 00|2E 00 00 00|EA BE 73 04|Ã!..........ê¾s■
5C 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|\...............
C6 21 00 00|03 00 00 00|E0 01 00 00|84 DF 5D 2E|Æ!..■...à■..■ß].
```

Primary configuration file

Number of sectors

Sector = 0x1000000 - 21C6 - 1

Size of data

Points to secondary file

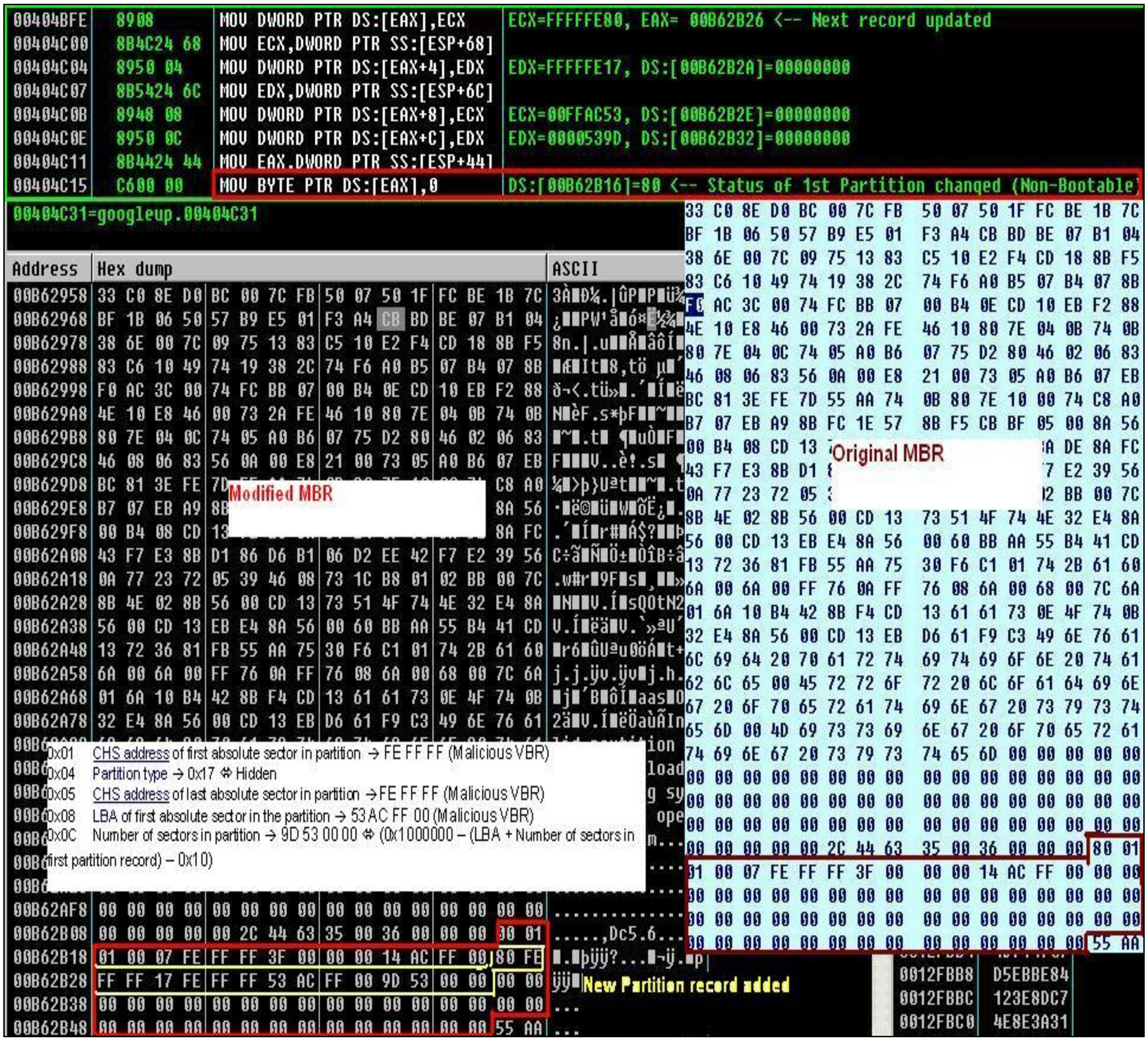
```
6D 62 72 00|00 00 00 00|00 00 00 00|00 00 00 00|mbr.............
C7 21 00 00|01 00 00 00|00 02 00 00|10 4F B3 52|Ç!..■....■..■O³R
76 62 72 00|00 00 00 00|00 00 00 00|00 00 00 00|vbr.............
C8 21 00 00|01 00 00 00|00 02 00 00|5F C5 B2 17|È!..■....■.._Å²■
62 69 64 00|00 00 00 00|00 00 00 00|00 00 00 00|bid.............
C9 21 00 00|01 00 00 00|25 00 00 00|9E 63 F4 A6|É!..■...%...■cô¦
61 66 66 69|64 00 00 00|00 00 00 00|00 00 00 00|affid...........
CA 21 00 00|01 00 00 00|04 00 00 00|6D D7 BF 7C|Ê!..■...■...m×¿|
62 6F 6F 74|00 00 00 00|00 00 00 00|00 00 00 00|boot............
CD 21 00 00|03 00 00 00|CF 05 00 00|E3 E5 83 CA|Í!..■...Ï■..ãå■Ê
63 6D 64 33|32 00 00 00|00 00 00 00|00 00 00 00|cmd32...........
0D 22 00 00|40 00 00 00|00 80 00 00|82 2E 01 63|."..@....■..■.■c
63 6D 64 36|34 00 00 00|00 00 00 00|00 00 00 00|cmd64...........
46 22 00 00|39 00 00 00|00 72 00 00|D2 DC 46 D8|F"..9....r..ÒÜFØ
64 62 67 33|32 00 00 00|00 00 00 00|00 00 00 00|dbg32...........
53 22 00 00|0D 00 00 00|00 1A 00 00|22 7F 74 24|S".......■.."■t$
64 62 67 36|34 00 00 00|00 00 00 00|00 00 00 00|dbg64...........
65 22 00 00|12 00 00 00|80 23 00 00|9C 10 E0 06|e"..■...■#..■■à■
64 72 76 33|32 00 00 00|00 00 00 00|00 00 00 00|drv32...........
B0 22 00 00|4B 00 00 00|00 96 00 00|4B 31 47 53|°"..K....■..K1GS
64 72 76 36|34 00 00 00|00 00 00 00|00 00 00 00|drv64...........
FB 22 00 00|4B 00 00 00|00 96 00 00|4B 3A 60 05|û"..K....■..K:`■
6C 64 72 33|32 00 00 00|00 00 00 00|00 00 00 00|ldr32...........
07 23 00 00|0C 00 00 00|00 18 00 00|4C 6D 39 06|■#.......■..Lm9■
6C 64 72 36|34 00 00 00|00 00 00 00|00 00 00 00|ldr64...........
12 23 00 00|0B 00 00 00|00 16 00 00|F8 BE D6 B8|■#..■....■..ø¾Ö¸
6D 61 69 6E|00 00 00 00|00 00 00 00|00 00 00 00|main............
17 23 00 00|05 00 00 00|AD 09 00 00|35 EB 07 55|■#..■...-...5ë■U
73 75 62 69|64 00 00 00|00 00 00 00|00 00 00 00|subid...........
18 23 00 00|01 00 00 00|08 00 00 00|BB 0F 22 ED|■#..■...■...»■"í
```

Secondary configuration file

The secondary configuration file is encrypted and saved into the hidden filesystem. Information about the sector where the secondary configuration file is saved with the number of sectors and data size is stored in the primary configuration file as shown in the above picture. Later the primary configuration file is encrypted, written to the disk and storage information is saved in the malicious VBR.

The malicious VBR is written into the first (0xFFAC53) and the last sectors (0xFFFFFE) of the partition.

Then the partition table in memory is updated to reflect the newly created partition and the new volume is made the bootable in order to transfer control to the malicious code as soon as possible as shown below:



```
00404BFE   8908         MOV DWORD PTR DS:[EAX],ECX     ECX=FFFFFE80, EAX= 00B62B26 <-- Next record updated
00404C00   8B4C24 68    MOV ECX,DWORD PTR SS:[ESP+68]
00404C04   8950 04      MOV DWORD PTR DS:[EAX+4],EDX   EDX=FFFFFE17, DS:[00B62B2A]=00000000
00404C07   8B5424 6C    MOV EDX,DWORD PTR SS:[ESP+6C]
00404C0B   8948 08      MOV DWORD PTR DS:[EAX+8],ECX   ECX=00FFAC53, DS:[00B62B2E]=00000000
00404C0E   8950 0C      MOV DWORD PTR DS:[EAX+C],EDX   EDX=0000539D, DS:[00B62B32]=00000000
00404C11   8B4424 44    MOV EAX,DWORD PTR SS:[ESP+44]
00404C15   C600 00      MOV BYTE PTR DS:[EAX],0        DS:[00B62B16]=80 <-- Status of 1st Partition changed (Non-Bootable)
00404C31=googleup.00404C31
```

Modified MBR

```
Address   Hex dump
00B62958  33 C0 8E D0 BC 00 7C FB 50 07 50 1F FC BE 1B 7C
00B62968  BF 1B 06 50 57 B9 E5 01 F3 A4 CB BD BE 07 B1 04
00B62978  38 6E 00 7C 09 75 13 83 C5 10 E2 F4 CD 18 8B F5
00B62988  83 C6 10 49 74 19 38 2C 74 F6 A0 B5 07 B4 07 8B
00B62998  F0 AC 3C 00 74 FC BB 07 00 B4 0E CD 10 EB F2 88
00B629A8  4E 10 E8 46 00 73 2A FE 46 10 80 7E 04 0B 74 0B
00B629B8  80 7E 04 0C 74 05 A0 B6 07 75 D2 80 46 02 06 83
00B629C8  46 08 06 83 56 0A 00 E8 21 00 73 05 A0 B6 07 EB
00B629D8  BC 81 3E FE 7D                            C8 A0
00B629E8  B7 07 EB A9 8B                            8A 56
00B629F8  00 B4 08 CD 13                            8A FC
00B62A08  43 F7 E3 8B D1 86 D6 B1 06 D2 EE 42 F7 E2 39 56
00B62A18  0A 77 23 72 05 39 46 08 73 1C B8 01 02 BB 00 7C
00B62A28  8B 4E 02 8B 56 00 CD 13 73 51 4F 74 4E 32 E4 8A
00B62A38  56 00 CD 13 EB E4 8A 56 00 60 BB AA 55 B4 41 CD
00B62A48  13 72 36 81 FB 55 AA 75 30 F6 C1 01 74 2B 61 60
00B62A58  6A 00 6A 00 FF 76 0A FF 76 08 6A 00 68 00 7C 6A
00B62A68  01 6A 10 B4 42 8B F4 CD 13 61 61 73 0E 4F 74 0B
00B62A78  32 E4 8A 56 00 CD 13 EB D6 61 F9 C3 49 6E 76 61
00B62AF8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B62B08  00 00 00 00 00 2C 44 63 35 00 36 00 00 00 00 01
00B62B18  01 00 07 FE FF FF 3F 00 00 00 14 AC FF 00 80 FE
00B62B28  FF FF 17 FE FF FF 53 AC FF 00 9D 53 00 00 00 00
00B62B38  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B62B48  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
```

0x01 CHS address of first absolute sector in partition → FE FF FF (Malicious VBR)
0x04 Partition type → 0x17 ⇔ Hidden
0x05 CHS address of last absolute sector in partition → FE FF FF (Malicious VBR)
0x08 LBA of first absolute sector in the partition → 53 AC FF 00 (Malicious VBR)
0x0C Number of sectors in partition → 9D 53 00 00 ⇔ (0x1000000 – (LBA + Number of sectors in first partition record) – 0x10)

New Partition record added

Original MBR

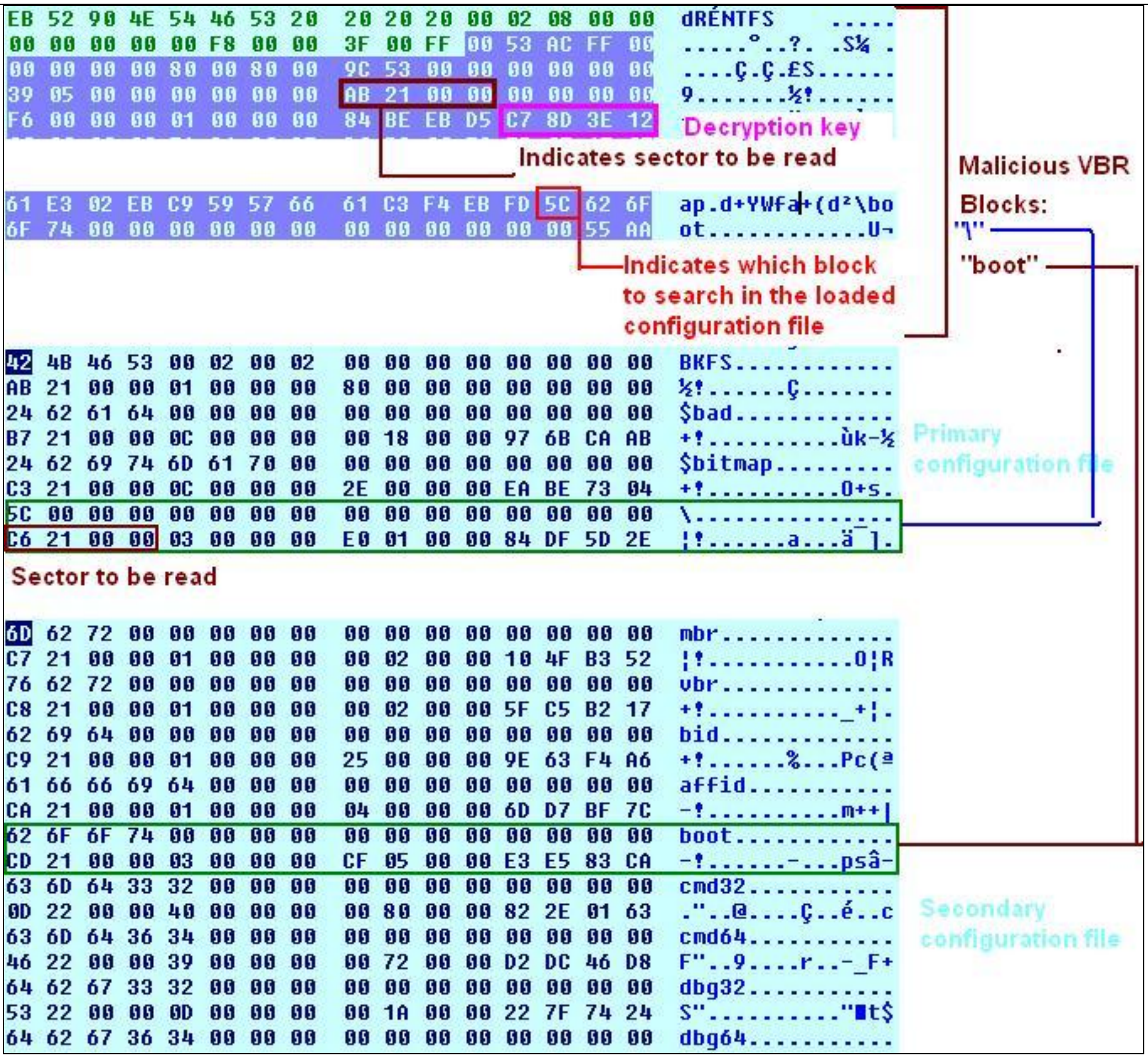
```
33 C0 8E D0 BC 00 7C FB  50 07 50 1F FC BE 1B 7C
BF 1B 06 50 57 B9 E5 01  F3 A4 CB BD BE 07 B1 04
38 6E 00 7C 09 75 13 83  C5 10 E2 F4 CD 18 8B F5
83 C6 10 49 74 19 38 2C  74 F6 A0 B5 07 B4 07 8B
F0 AC 3C 00 74 FC BB 07  00 B4 0E CD 10 EB F2 88
4E 10 E8 46 00 73 2A FE  46 10 80 7E 04 0B 74 0B
80 7E 04 0C 74 05 A0 B6  07 75 D2 80 46 02 06 83
46 08 06 83 56 0A 00 E8  21 00 73 05 A0 B6 07 EB
BC 81 3E FE 7D 55 AA 74  0B 80 7E 10 00 74 C8 A0
B7 07 EB A9 8B FC 1E 57  8B F5 CB BF 05 00 8A 56
00 B4 08 CD 13                      DE 8A FC
43 F7 E3 8B D1                       7 E2 39 56
0A 77 23 72 05                       2 BB 00 7C
8B 4E 02 8B 56 00 CD 13  73 51 4F 74 4E 32 E4 8A
56 00 CD 13 EB E4 8A 56  00 60 BB AA 55 B4 41 CD
13 72 36 81 FB 55 AA 75  30 F6 C1 01 74 2B 61 60
6A 00 6A 00 FF 76 0A FF  76 08 6A 00 68 00 7C 6A
01 6A 10 B4 42 8B F4 CD  13 61 61 73 0E 4F 74 0B
32 E4 8A 56 00 CD 13 EB  D6 61 F9 C3 49 6E 76 61
6C 69 64 20 70 61 72 74  69 74 69 6F 6E 20 74 61
62 6C 65 00 45 72 72 6F  72 20 6C 6F 61 64 69 6E
67 20 6F 70 65 72 61 74  69 6E 67 20 73 79 73 74
65 6D 00 4D 69 73 73 69  6E 67 20 6F 70 65 72 61
74 69 6E 67 20 73 79 73  74 65 6D 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 2C 44 63  35 00 36 00 00 00 80 01
01 00 07 FE FF FF 3F 00  00 00 14 AC FF 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 AA
```

```
0012FBB8  D5EBBE84
0012FBBC  123E8DC7
0012FBC0  4E8E3A31
```

(1.12 Partition table modified)

Finally, the malicious dropper file is deleted from the infected system to remove traces of infection. It also mark files in the %TEMP% folder for deletion on reboot and the system is rebooted.

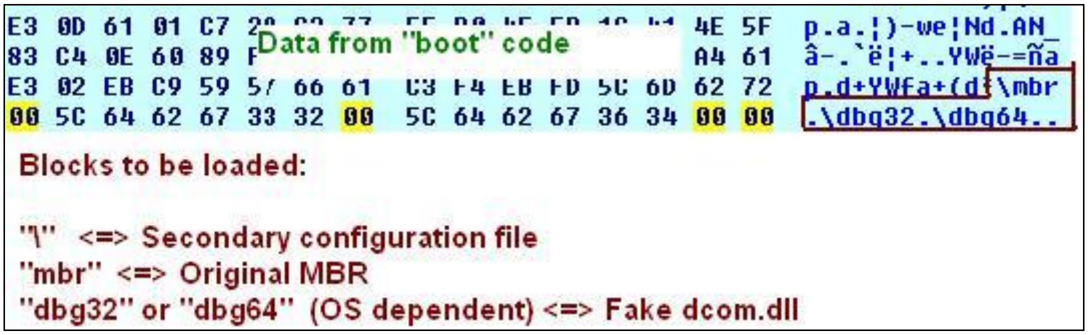
On reboot, malicious VBR residing in the hidden file system is loaded by the MBR and control is transferred to the VBR code. It first reads the sector containing primary configuration file which is later parsed to retrieve information about the secondary configuration file. The secondary configuration file is then parsed to load sectors (block named “boot” in the secondary configuration file) containing code which is responsible for hooking Interrupt Vector Table as shown below:



Decryption key
Indicates sector to be read
Malicious VBR
Blocks:
"\"
"boot"
Indicates which block to search in the loaded configuration file
Primary configuration file
Sector to be read
Secondary configuration file

```
EB 52 90 4E 54 46 53 20  20 20 20 00 02 08 00 00  dRÉNTFS    .....
00 00 00 00 00 F8 00 00  3F 00 FF 00 53 AC FF 00  .....°..?. .S¼ .
00 00 00 00 80 00 80 00  9C 53 00 00 00 00 00 00  ....Ç.Ç.£S......
39 05 00 00 00 00 00 00  AB 21 00 00 00 00 00 00  9.......½!......
F6 00 00 00 01 00 00 00  84 BE EB D5 C7 8D 3E 12

61 E3 02 EB C9 59 57 66  61 C3 F4 EB FD 5C 62 6F  ap.d+YWfa+(d²\bo
6F 74 00 00 00 00 00 00  00 00 00 00 00 00 55 AA  ot............U¬

42 4B 46 53 00 02 00 02  00 00 00 00 00 00 00 00  BKFS............
AB 21 00 00 01 00 00 00  80 00 00 00 00 00 00 00  ½!......Ç.......
24 62 61 64 00 00 00 00  00 00 00 00 00 00 00 00  $bad............
B7 21 00 00 0C 00 00 00  00 18 00 00 97 6B CA AB  +!..........ùk-½
24 62 69 74 6D 61 70 00  00 00 00 00 00 00 00 00  $bitmap.........
C3 21 00 00 0C 00 00 00  2E 00 00 00 EA BE 73 04  +!..........O+s.
5C 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  \...............
C6 21 00 00 03 00 00 00  E0 01 00 00 84 DF 5D 2E  ¦!......a...ä¯].

6D 62 72 00 00 00 00 00  00 00 00 00 00 00 00 00  mbr.............
C7 21 00 00 01 00 00 00  00 02 00 00 10 4F B3 52  ¦!...........O¦R
76 62 72 00 00 00 00 00  00 00 00 00 00 00 00 00  vbr.............
C8 21 00 00 01 00 00 00  00 02 00 00 5F C5 B2 17  +!.........._+¦.
62 69 64 00 00 00 00 00  00 00 00 00 00 00 00 00  bid.............
C9 21 00 00 01 00 00 00  25 00 00 00 9E 63 F4 A6  +!......%...Pc(ª
61 66 66 69 64 00 00 00  00 00 00 00 00 00 00 00  affid...........
CA 21 00 00 01 00 00 00  04 00 00 00 6D D7 BF 7C  -!..........m++|
62 6F 6F 74 00 00 00 00  00 00 00 00 00 00 00 00  boot............
CD 21 00 00 03 00 00 00  CF 05 00 00 E3 E5 83 CA  -!......-...psâ-
63 6D 64 33 32 00 00 00  00 00 00 00 00 00 00 00  cmd32...........
0D 22 00 00 40 00 00 00  00 80 00 00 82 2E 01 63  ."..@....Ç..é..c
63 6D 64 36 34 00 00 00  00 00 00 00 00 00 00 00  cmd64...........
46 22 00 00 39 00 00 00  00 72 00 00 D2 DC 46 D8  F"..9....r..-_F+
64 62 67 33 32 00 00 00  00 00 00 00 00 00 00 00  dbg32...........
53 22 00 00 0D 00 00 00  00 1A 00 00 22 7F 74 24  S"..........".t$
64 62 67 36 34 00 00 00  00 00 00 00 00 00 00 00  dbg64...........
```

(1.13 Picture shows how VBR decides what to load)



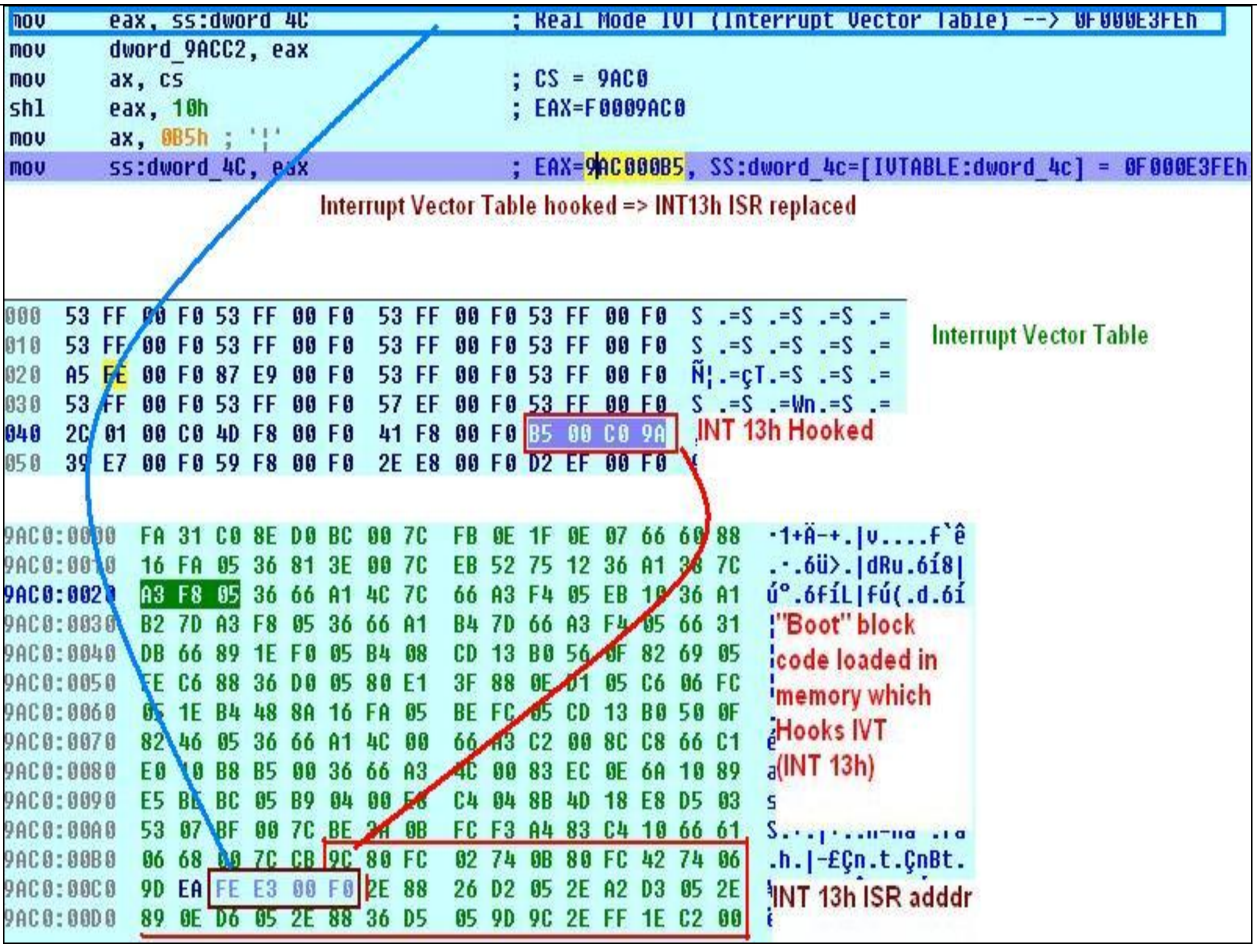
```
E3 0D 61 01 C7 2                             4E 5F  p.a.¦)-we¦Nd.AN_
83 C4 0E 60 89 F                             A4 61  â-.`ë¦+..YWë-=ña
E3 02 EB C9 59 57 66 61  C3 F4 EB FD 5C 6D 62 72  p.d+YWfa+(d²\mbr
00 5C 64 62 67 33 32 00  5C 64 62 67 36 34 00 00  .\dbg32.\dbg64..
```

Blocks to be loaded:

"\" <=> Secondary configuration file
"mbr" <=> Original MBR
"dbg32" or "dbg64" (OS dependent) <=> Fake dcom.dll

(1.14 Picture shows how boot code decides what to load)

Interrupt Vector Table (IVT) Hook
Pointer to the Interrupt Service Routine (ISR) INT13h is replaced with an address which points to the malware's code.

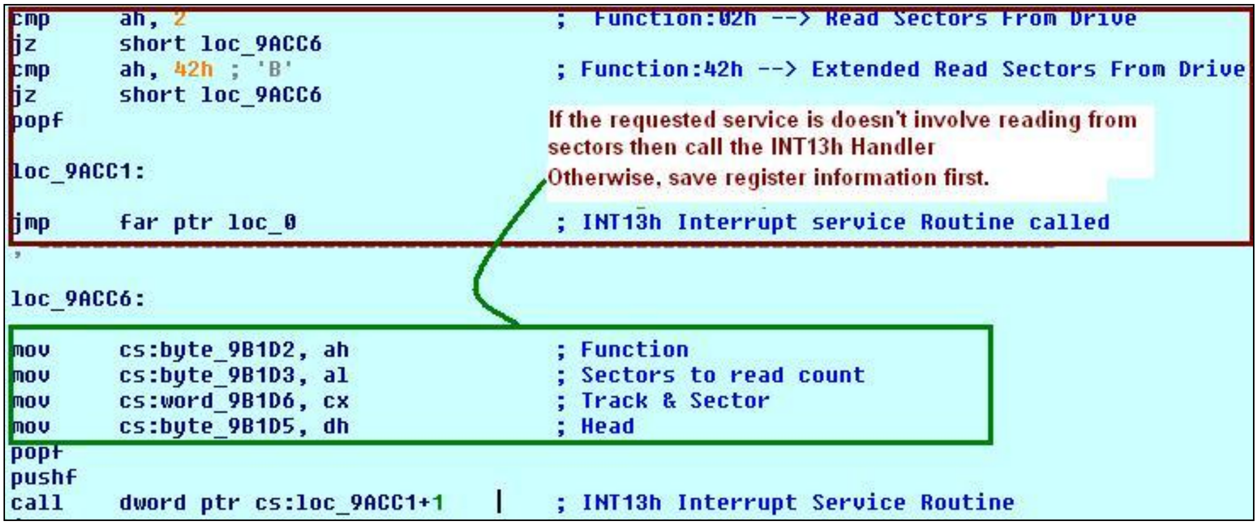


```
mov     eax, ss:dword_4C                ; Real Mode IVT (Interrupt Vector Table) --> 0F000E3FEh
mov     dword_9ACC2, eax
mov     ax, cs                          ; CS = 9AC0
shl     eax, 10h                        ; EAX=F0009AC0
mov     ax, 0B5h ; '¦'
mov     ss:dword_4C, eax                ; EAX=9AC000B5, SS:dword_4c=[IVTABLE:dword_4c] = 0F000E3FEh

000  53 FF 00 F0 53 FF 00 F0  53 FF 00 F0 53 FF 00 F0  S .=S .=S .=S .=
010  53 FF 00 F0 53 FF 00 F0  53 FF 00 F0 53 FF 00 F0  S .=S .=S .=S .=
020  A5 FE 00 F0 87 E9 00 F0  53 FF 00 F0 53 FF 00 F0  Ñ¦.=çT.=S .=S .=
030  53 FF 00 F0 53 FF 00 F0  57 EF 00 F0 53 FF 00 F0  S .=S .=Wn.=S .=
040  2C 01 00 C0 4D F8 00 F0  41 F8 00 F0 B5 00 C0 9A
050  39 E7 00 F0 59 F8 00 F0  2E E8 00 F0 D2 EF 00 F0

9AC0:0000  FA 31 C0 8E D0 BC 00 7C  FB 0E 1F 0E 07 66 60 88  ·1+Ä-+.|v....f`ê
9AC0:0010  16 FA 05 36 81 3E 00 7C  EB 52 75 12 36 A1 38 7C  .·.6ü>.|dRu.6í8|
9AC0:0020  A3 F8 05 36 66 A1 4C 7C  66 A3 F4 05 EB 10 36 A1  ú°.6fíL|fú(.d.6í
9AC0:0030  B2 7D A3 F8 05 36 66 A1  B4 7D 66 A3 F4 05 66 31
9AC0:0040  DB 66 89 1E F0 05 B4 08  CD 13 B0 56 0F 82 69 05
9AC0:0050  FE C6 88 36 D0 05 80 E1  3F 88 0E D1 05 C6 06 FC
9AC0:0060  05 1E B4 48 8A 16 FA 05  BE FC 05 CD 13 B0 50 0F
9AC0:0070  82 46 05 36 66 A1 4C 00  66 A3 C2 00 8C C8 66 C1  é
9AC0:0080  E0 10 B8 B5 00 36 66 A3  4C 00 83 EC 0E 6A 10 89  a
9AC0:0090  E5 BE BC 05 B9 04 00 E8  C4 04 8B 4D 18 E8 D5 03  s
9AC0:00A0  53 07 BF 00 7C BE 3A 0B  FC F3 A4 83 C4 10 66 61  S..
9AC0:00B0  06 68 00 7C CB 9C 80 FC  02 74 0B 80 FC 42 74 06  .h.|-£Çn.t.ÇnBt.
9AC0:00C0  9D EA FE E3 00 F0 2E 88  26 D2 05 2E A2 D3 05 2E
9AC0:00D0  89 0E D6 05 2E 88 36 D5  05 9D 9C 2E FF 1E C2 00  é
```

Interrupt Vector Table hooked => INT13h ISR replaced
Interrupt Vector Table
INT 13h Hooked
"Boot" block code loaded in memory which Hooks IVT (INT 13h)
INT 13h ISR adddr

(1.15 Interrupt vector table hook installed)

INT13h hook checks which service is being requested. If the service doesn’t involve sector read operation (function code: 02h and 42h), it calls the original INT13h handler and transfers the control back to the caller as shown below:



```
cmp     ah, 2                           ;  Function:02h --> Read Sectors From Drive
jz      short loc_9ACC6
cmp     ah, 42h ; 'B'                   ; Function:42h --> Extended Read Sectors From Drive
jz      short loc_9ACC6
popf

loc_9ACC1:

jmp     far ptr loc_0                   ; INT13h Interrupt service Routine called
;

loc_9ACC6:

mov     cs:byte_9B1D2, ah               ; Function
mov     cs:byte_9B1D3, al               ; Sectors to read count
mov     cs:word_9B1D6, cx               ; Track & Sector
mov     cs:byte_9B1D5, dh               ; Head
popf
pushf
call    dword ptr cs:loc_9ACC1+1        ; INT13h Interrupt Service Routine
```

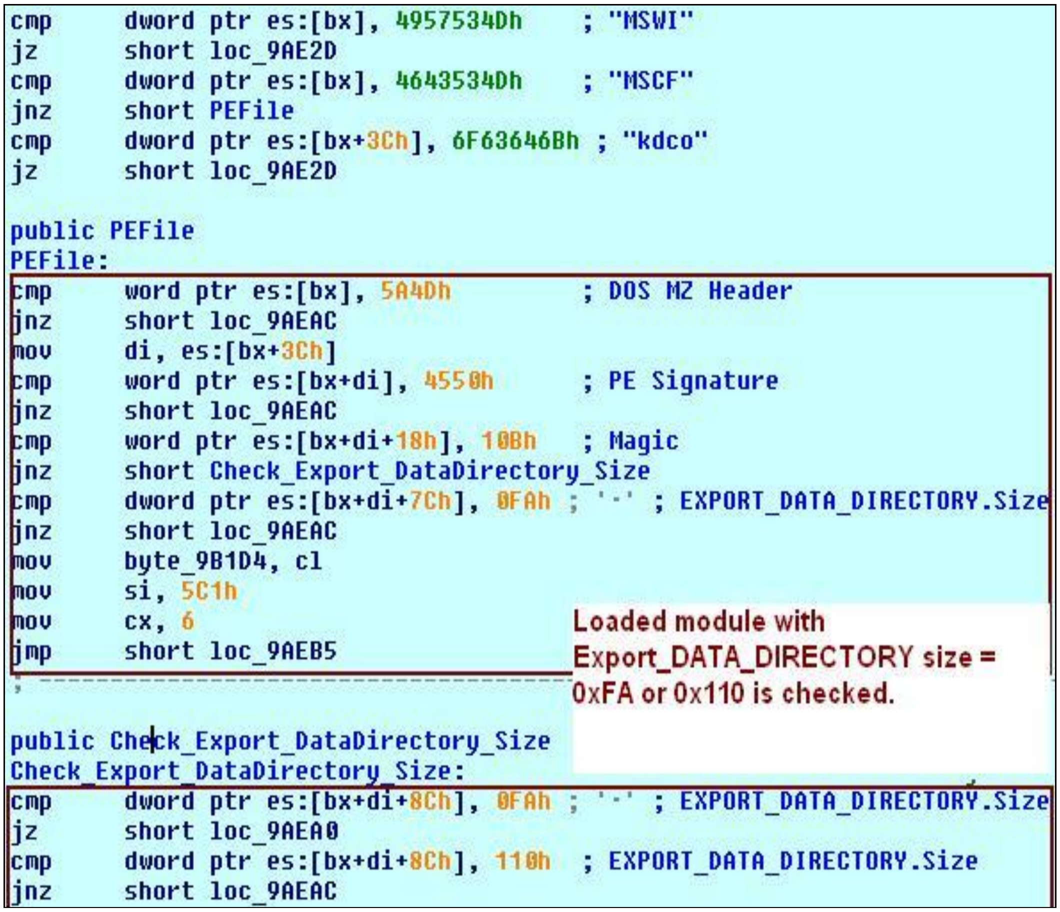
If the requested service is doesn't involve reading from sectors then call the INT13h Handler
Otherwise, save register information first.

(1.16 INT13h Services to monitor)

If the requested service involves sector read operation, it saves information like the number of sectors to read; sector number etc before calling the original INT13h handler. After reading the sector into memory, it checks if it matches either of the following conditions:

- PE Image with IMAGE_DIRECTORY_ENTRY_EXPORT.Size == 0xFA or 0x110
    - If it finds any module matching the above condition, it loads dbg32 or dbg64 (fake kdcom.dll) depending on the OS environment

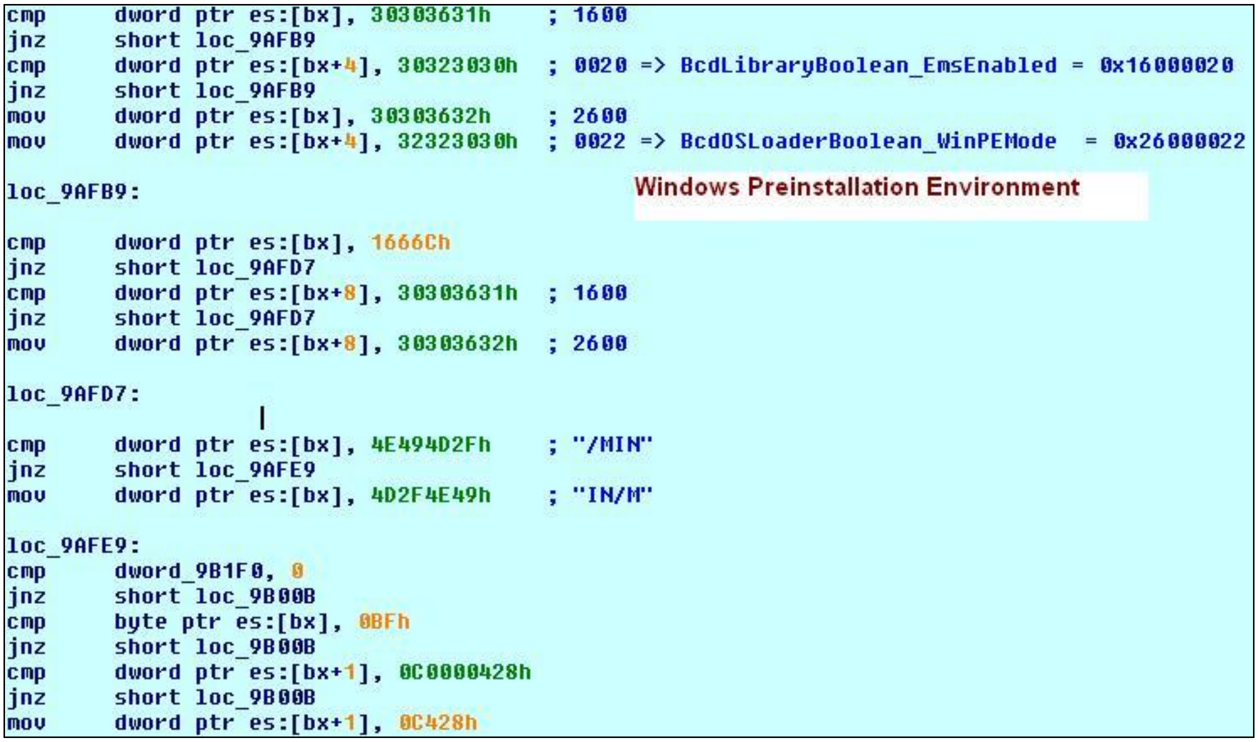
- Check Boot Configuration Data (BCD) store for BcdLibraryBoolean_EmsEnabled [16000020]. If it is found, replace it with BcdOsLoaderBoolean_WinPEMode [26000022].



```
cmp     dword ptr es:[bx], 4957534Dh    ; "MSWI"
jz      short loc_9AE2D
cmp     dword ptr es:[bx], 4643534Dh    ; "MSCF"
jnz     short PEFile
cmp     dword ptr es:[bx+3Ch], 6F63646Bh ; "kdco"
jz      short loc_9AE2D

public PEFile
PEFile:
cmp     word ptr es:[bx], 5A4Dh         ; DOS MZ Header
jnz     short loc_9AEAC
mov     di, es:[bx+3Ch]
cmp     word ptr es:[bx+di], 4550h      ; PE Signature
jnz     short loc_9AEAC
cmp     word ptr es:[bx+di+18h], 10Bh   ; Magic
jnz     short Check_Export_DataDirectory_Size
cmp     dword ptr es:[bx+di+7Ch], 0FAh ; '·' ; EXPORT_DATA_DIRECTORY.Size
jnz     short loc_9AEAC
mov     byte_9B1D4, cl
mov     si, 5C1h
mov     cx, 6
jmp     short loc_9AEB5
; ---------------------------------------------------------------------------

public Check_Export_DataDirectory_Size
Check_Export_DataDirectory_Size:
cmp     dword ptr es:[bx+di+8Ch], 0FAh ; '·' ; EXPORT_DATA_DIRECTORY.Size
jz      short loc_9AEA0
cmp     dword ptr es:[bx+di+8Ch], 110h  ; EXPORT_DATA_DIRECTORY.Size
jnz     short loc_9AEAC
```

Loaded module with Export_DATA_DIRECTORY size = 0xFA or 0x110 is checked.

(1.17 Check module with IMAGE_DIRECTORY_ENTRY_ EXPORT size = 0xFA or 0x110)



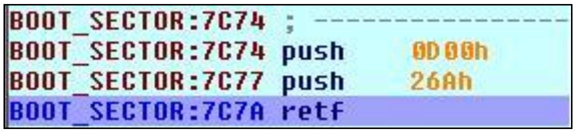
```
cmp     dword ptr es:[bx], 30303631h    ; 1600
jnz     short loc_9AFB9
cmp     dword ptr es:[bx+4], 30323030h  ; 0020 => BcdLibraryBoolean_EmsEnabled = 0x16000020
jnz     short loc_9AFB9
mov     dword ptr es:[bx], 30303632h    ; 2600
mov     dword ptr es:[bx+4], 32323030h  ; 0022 => BcdOSLoaderBoolean_WinPEMode   = 0x26000022

loc_9AFB9:

cmp     dword ptr es:[bx], 1666Ch
jnz     short loc_9AFD7
cmp     dword ptr es:[bx+8], 30303631h  ; 1600
jnz     short loc_9AFD7
mov     dword ptr es:[bx+8], 30303632h  ; 2600

loc_9AFD7:

cmp     dword ptr es:[bx], 4E494D2Fh    ; "/MIN"
jnz     short loc_9AFE9
mov     dword ptr es:[bx], 4D2F4E49h    ; "IN/M"

loc_9AFE9:
cmp     dword_9B1F0, 0
jnz     short loc_9B00B
cmp     byte ptr es:[bx], 0BFh
jnz     short loc_9B00B
cmp     dword ptr es:[bx+1], 0C0000428h
jnz     short loc_9B00B
mov     dword ptr es:[bx+1], 0C428h
```

Windows Preinstallation Environment

(1.18 Windows Pre-installation Environment)

After installing Interrupt Vector Table hook, it again reads sectors containing primary and secondary configuration file. Then it looks for a block named “**mbr**” in the secondary configuration file which contains original MBR.

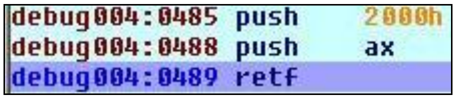
Now the original MBR gets control, it loads the VBR and transfers control to it. VBR boot code loads the bootstrap code (0xF Sectors following the VBR). It first loads itself at address 0D00:0000, then following 0xF sectors are loaded in the successive memory addresses. Once all the sectors have been loaded, it transfers control to the bootstrap code as shown below:



```
BOOT_SECTOR:7C74 ; ----------------
BOOT_SECTOR:7C74 push    0D00h
BOOT_SECTOR:7C77 push    26Ah
BOOT_SECTOR:7C7A retf
```

(1.19 Control transferred to bootstrap code)

Bootstrap code read contents from the root drive, loads NTLDR at address 2000:0000h and transfers control to the NTLDR.



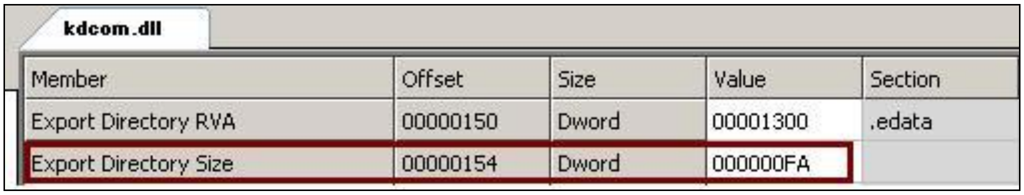
```
debug004:0485 push    2000h
debug004:0488 push    ax
debug004:0489 retf
```

(1.20 Control transferred to NTLDR)

NTLDR contains an embedded PE file (osloader.exe) which loads the Windows system files (starting with the ntoskrnl.exe, its dependencies (HAL.dll, bootvid.dll, and kdcom.dll), SYSTEM hive, and the boot drivers) into memory.

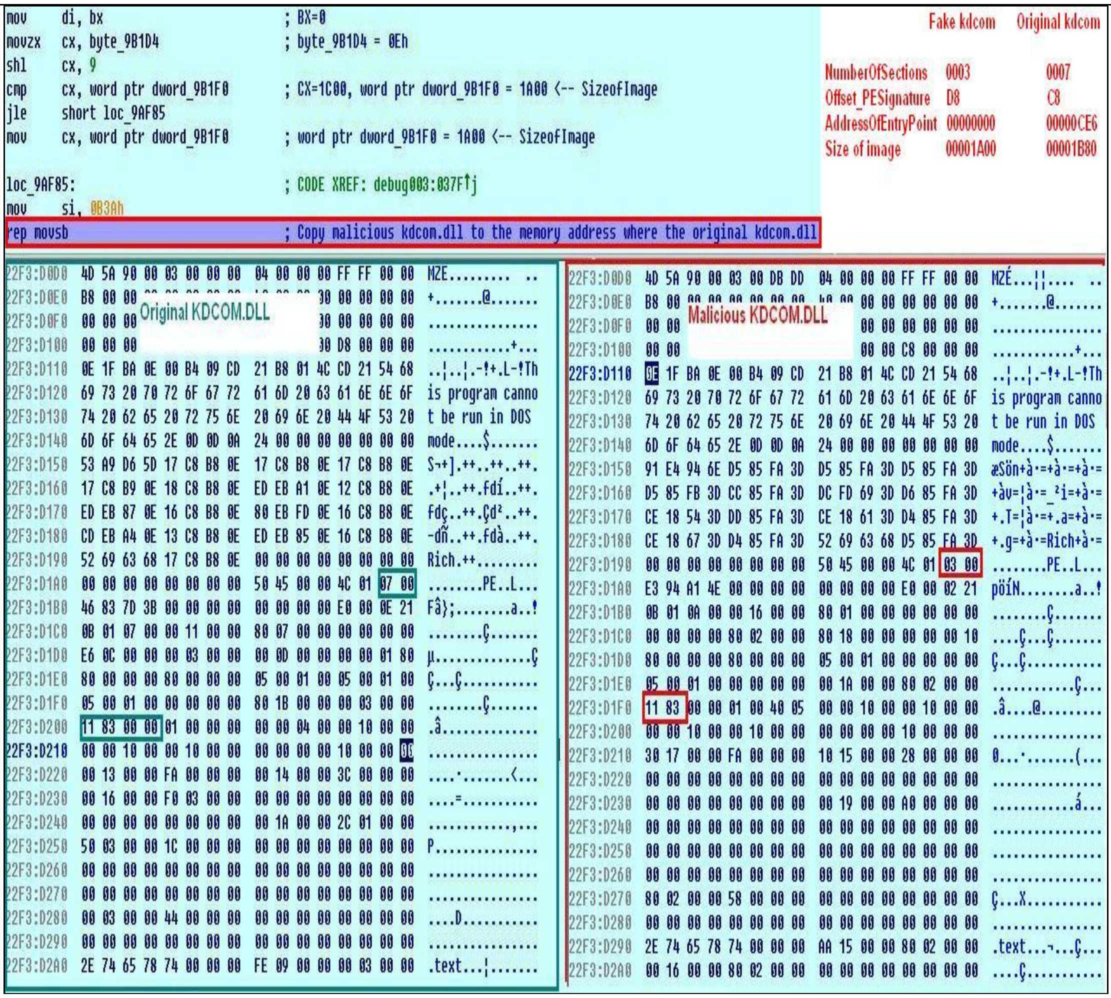
osloader.exe mostly executes in protected mode, but for input/output operation it depends on the BIOS services. So it keeps switching between real and protected mode.

Since the size of kdocm.dll’s EXPORT_DATA_DIRECTORY is 0xFA. As soon as kdcom.dll is loaded, INT13h hook loads fake kdcom.dll (dbg32 or dbg64) into memory, updates checksum and replaces the original kdcom.dll with the fake kdcom.dll as shown below. Control is then transferred back to the osloader.exe



Member	Offset	Size	Value	Section
Export Directory RVA	00000150	Dword	00001300	.edata
Export Directory Size	00000154	Dword	000000FA	

(1.21 kdcom.dll IMAGE_DIRECTORY_ENTRY_EXPORT.Size = 0xFA)



```
mov     di, bx                          ; BX=0
movzx   cx, byte_9B1D4                  ; byte_9B1D4 = 0Eh
shl     cx, 9
cmp     cx, word ptr dword_9B1F0        ; CX=1C00, word ptr dword_9B1F0 = 1A00 <-- SizeofImage
jle     short loc_9AF85
mov     cx, word ptr dword_9B1F0        ; word ptr dword_9B1F0 = 1A00 <-- SizeofImage

loc_9AF85:                              ; CODE XREF: debug003:037F↑j
mov     si, 0B3Ah
rep movsb                               ; Copy malicious kdcom.dll to the memory address where the original kdcom.dll
```

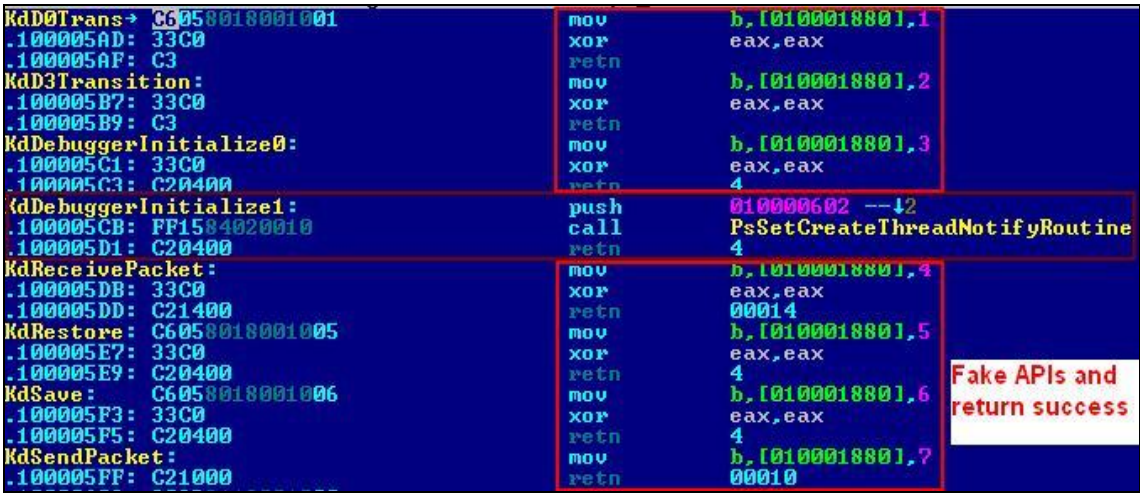
	Fake kdcom	Original kdcom
NumberOfSections	0003	0007
Offset_PESignature	D8	C8
AddressOfEntryPoint	00000000	00000CE6
Size of image	00001A00	00001B80

```
22F3:D0D0  4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZE.........  ..
22F3:D0E0  B8 00 00                             00 00 00 00  +.......@.......
22F3:D0F0  00 00 00                             00 00 00 00  ................
22F3:D100  00 00 00                             D8 00 00 00  ............+...
22F3:D110  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ..¦..¦.-!+.L-!Th
22F3:D120  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
22F3:D130  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS
22F3:D140  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode....$.......
22F3:D150  53 A9 D6 5D 17 C8 B8 0E  17 C8 B8 0E 17 C8 B8 0E  S¬+].++..++..++.
22F3:D160  17 C8 B9 0E 18 C8 B8 0E  ED EB A1 0E 12 C8 B8 0E  .+¦..++.fdí..++.
22F3:D170  ED EB 87 0E 16 C8 B8 0E  80 EB FD 0E 16 C8 B8 0E  fdç..++.Çd²..++.
22F3:D180  CD EB A4 0E 13 C8 B8 0E  ED EB 85 0E 16 C8 B8 0E  -dñ..++.fdà..++.
22F3:D190  52 69 63 68 17 C8 B8 0E  00 00 00 00 00 00 00 00  Rich.++.........
22F3:D1A0  00 00 00 00 00 00 00 00  50 45 00 00 4C 01 07 00  ........PE..L...
22F3:D1B0  46 83 7D 3B 00 00 00 00  00 00 00 00 E0 00 0E 21  Fâ};........a..!
22F3:D1C0  0B 01 07 00 00 11 00 00  80 07 00 00 00 00 00 00  ........Ç.......
22F3:D1D0  E6 0C 00 00 00 03 00 00  00 0D 00 00 00 00 01 80  µ..............Ç
22F3:D1E0  80 00 00 00 80 00 00 00  05 00 01 00 05 00 01 00  Ç...Ç...........
22F3:D1F0  05 00 01 00 00 00 00 00  80 1B 00 00 00 03 00 00  ........Ç.......
22F3:D200  11 83 00 00 01 00 00 00  00 00 04 00 00 10 00 00  .â..............
22F3:D210  00 00 10 00 00 10 00 00  00 00 00 00 10 00 00 00  ................
22F3:D220  00 13 00 00 FA 00 00 00  00 14 00 00 3C 00 00 00  ....·.......<...
22F3:D230  00 16 00 00 F0 03 00 00  00 00 00 00 00 00 00 00  ....=...........
22F3:D240  00 00 00 00 00 00 00 00  00 1A 00 00 2C 01 00 00  ............,...
22F3:D250  50 03 00 00 1C 00 00 00  00 00 00 00 00 00 00 00  P...............
22F3:D260  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
22F3:D270  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
22F3:D280  00 03 00 00 44 00 00 00  00 00 00 00 00 00 00 00  ....D...........
22F3:D290  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
22F3:D2A0  2E 74 65 78 74 00 00 00  FE 09 00 00 00 03 00 00  .text...¦.......
```

```
22F3:D0D0  4D 5A 90 00 03 00 DB DD  04 00 00 00 FF FF 00 00  MZÉ...¦¦....  ..
22F3:D0E0  B8 00                          00 00 00 00 00 00  +.......@.......
22F3:D0F0  00 00                          00 00 00 00 00 00  ................
22F3:D100  00 00                          00 00 C8 00 00 00  ............+...
22F3:D110  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ..¦..¦.-!+.L-!Th
22F3:D120  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
22F3:D130  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS
22F3:D140  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode....$.......
22F3:D150  91 E4 94 6E D5 85 FA 3D  D5 85 FA 3D D5 85 FA 3D  æSön+à·=+à·=+à·=
22F3:D160  D5 85 FB 3D CC 85 FA 3D  DC FD 69 3D D6 85 FA 3D  +àv=¦à·=_²i=+à·=
22F3:D170  CE 18 54 3D DD 85 FA 3D  CE 18 61 3D D4 85 FA 3D  +.T=¦à·=+.a=+à·=
22F3:D180  CE 18 67 3D D4 85 FA 3D  52 69 63 68 D5 85 FA 3D  +.g=+à·=Rich+à·=
22F3:D190  00 00 00 00 00 00 00 00  50 45 00 00 4C 01 03 00  ........PE..L...
22F3:D1A0  E3 94 A1 4E 00 00 00 00  00 00 00 00 E0 00 02 21  pöíN........a..!
22F3:D1B0  0B 01 0A 00 00 16 00 00  80 01 00 00 00 00 00 00  ........Ç.......
22F3:D1C0  00 00 00 00 80 02 00 00  80 18 00 00 00 00 00 10  ....Ç...Ç.......
22F3:D1D0  80 00 00 00 80 00 00 00  05 00 01 00 00 00 00 00  Ç...Ç...........
22F3:D1E0  05 00 01 00 00 00 00 00  00 1A 00 00 80 02 00 00  ............Ç...
22F3:D1F0  11 83 00 00 01 00 40 05  00 00 10 00 00 10 00 00  .â....@.........
22F3:D200  00 00 10 00 00 10 00 00  00 00 00 00 10 00 00 00  ................
22F3:D210  30 17 00 00 FA 00 00 00  10 15 00 00 28 00 00 00  0...·.......(...
22F3:D220  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
22F3:D230  00 00 00 00 00 00 00 00  00 19 00 00 A0 00 00 00  ............á...
22F3:D240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
22F3:D250  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
22F3:D260  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
22F3:D270  80 02 00 00 58 00 00 00  00 00 00 00 00 00 00 00  Ç...X...........
22F3:D280  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
22F3:D290  2E 74 65 78 74 00 00 00  AA 15 00 00 80 02 00 00  .text...¬...Ç...
22F3:D2A0  00 16 00 00 80 02 00 00  00 00 00 00 00 00 00 00  ....Ç...........
```

(1.22 Original kdcom.dll is replaced in memory with dbg32 – fake kdcom.dll)

KDCOM.DLL is COM-based debugging plug-in, so by faking the exported APIs it is actually disabling kernel debugging option via COM port.



```
KdD0Trans→ C605801800l001         mov    b,[010001880],1
.100005AD: 33C0                   xor    eax,eax
.100005AF: C3                     retn
KdD3Transition:                   mov    b,[010001880],2
.100005B7: 33C0                   xor    eax,eax
.100005B9: C3                     retn
KdDebuggerInitialize0:            mov    b,[010001880],3
.100005C1: 33C0                   xor    eax,eax
.100005C3: C20400                 retn   4
KdDebuggerInitialize1:            push   010000602 --↓2
.100005CB: FF1584020010           call   PsSetCreateThreadNotifyRoutine
.100005D1: C20400                 retn   4
KdReceivePacket:                  mov    b,[010001880],4
.100005DB: 33C0                   xor    eax,eax
.100005DD: C21400                 retn   00014
KdRestore: C605801800l005         mov    b,[010001880],5
.100005E7: 33C0                   xor    eax,eax
.100005E9: C20400                 retn   4
KdSave:    C605801800l006         mov    b,[010001880],6
.100005F3: 33C0                   xor    eax,eax
.100005F5: C20400                 retn   4
KdSendPacket:                     mov    b,[010001880],7
.100005FF: C21000                 retn   00010
```

(1.23 APIs faked by malicious kdcom.dll)

Mitigation

- Block access to the unused ports and block the access to the above mentioned URLs.
- Users who are identified to be infected are requested to change their passwords.

- Reboot the system in safe mode and log in as the Administrator user.

  Execute the CSSCAN command line tool using the Beta DATs to remove any Trojan or infected file from the system:

  - **VSE 8.7**
    "C:\Program Files\McAfee\VirusScan Enterprise\csscan.exe" -All -Unzip -Program -Analyze -Sub -Clean -Log c:\scan-rpt.txt C:\

  - **VSE 8.8**
    “C:\Program Files\Common Files\McAfee\SystemCore\csscan.exe” -All -Unzip -Program -Analyze -Sub -Clean -Log c:\scan-rpt.txt C:\

  - **Other McAfee product users**
    Please use the following Stinger standalone tool.

    To use the Stinger tool, please make sure the targets "Processes" and "Registry" are disabled and the interface "List of all files scanned" is enabled in the stinger before scanning the infected machine.

- Read more about using the Stinger tool here.

- Reboot the system normally.

- Run GMER again to confirm that no malicious threads of patched files exist anymore.

Restart Mechanism

The malware restarts by randomly infecting a system driver (usually located in %windir%/system32/drivers). This particular variant mostly infects the file VOLSNAP.SYS

Getting Help from the McAfee Foundstone Services team

This document is intended to provide a summary of current intelligence and best practices to ensure the highest level of protection from your McAfee security solution. The McAfee Foundstone Services team offers a full range of strategic and technical consulting services that can further help to ensure you identify security risk and build effective solutions to remediate security vulnerabilities.

You can reach them here: https://secure.mcafee.com/apps/services/services-contact.aspx