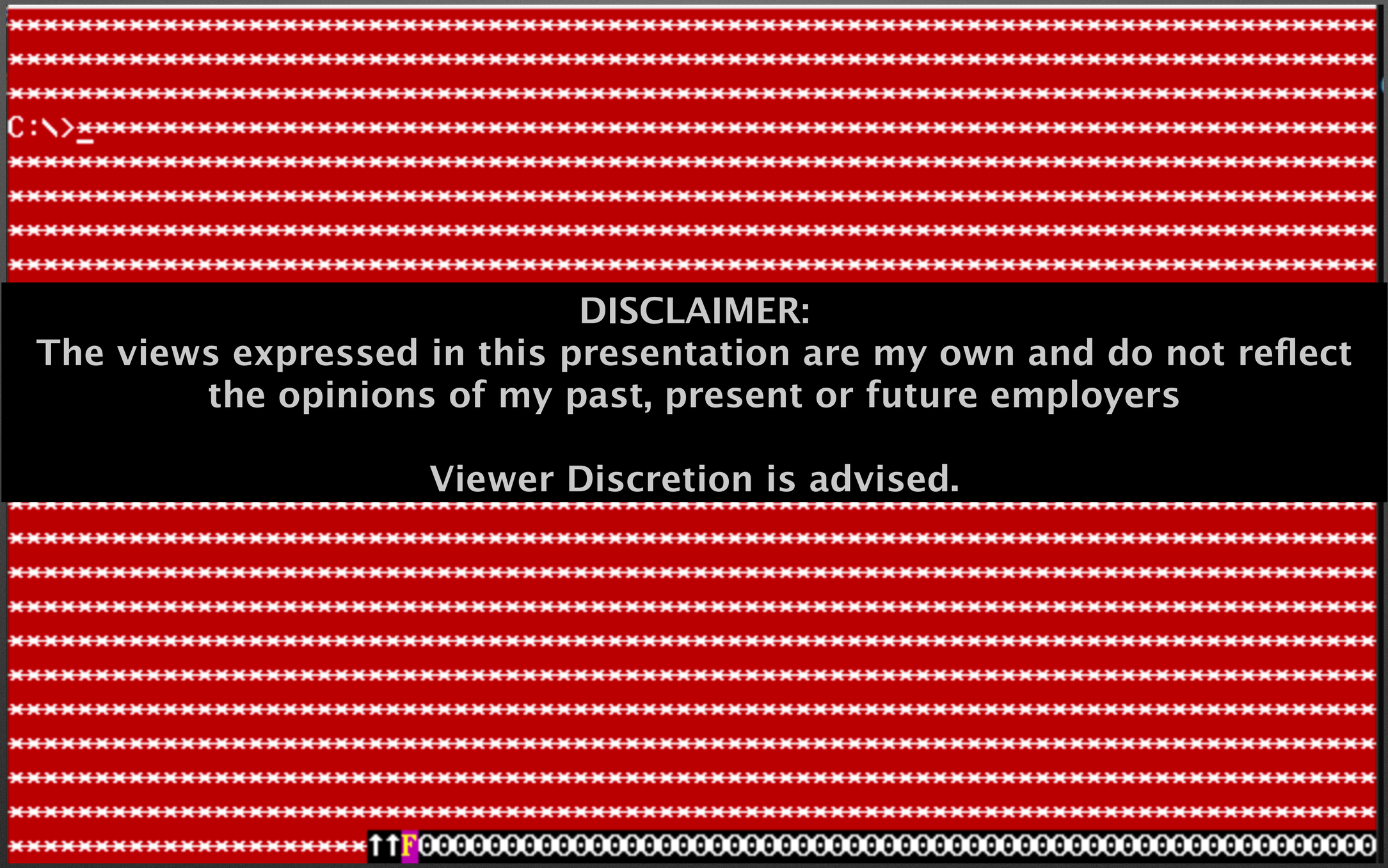

My Super Sweet 16-Bit Malware —
~*MS-DOS *~ Edition
Nika RSK V

whoami

Twitter: @nikaroxanne and @v1kt0r_frnknstn
Instagram: @nikaroxanne
Mastodon: ic3qu33n@infosec.exchange
Website: nikaroxanne.github.io
GitHub: @nikaroxanne
LinkedIn (?): linkedin.com/nika-kw

Application Security Engineer by day,
Reverse engineer + artist by night
I <3 malware, hardware hacking, skateboarding,
learning languages, creating art, writing lil assembly programs,
etc.

Shoutout to the following for their assistance/support w this talk:
Ben Mason (@suidroot), 0day_simpson,
Richard Johnson (@richinseattle (thank you for the invite!)),
Greg Linares (@Laughing_Mantis),
Hushcon



DISCLAIMER:

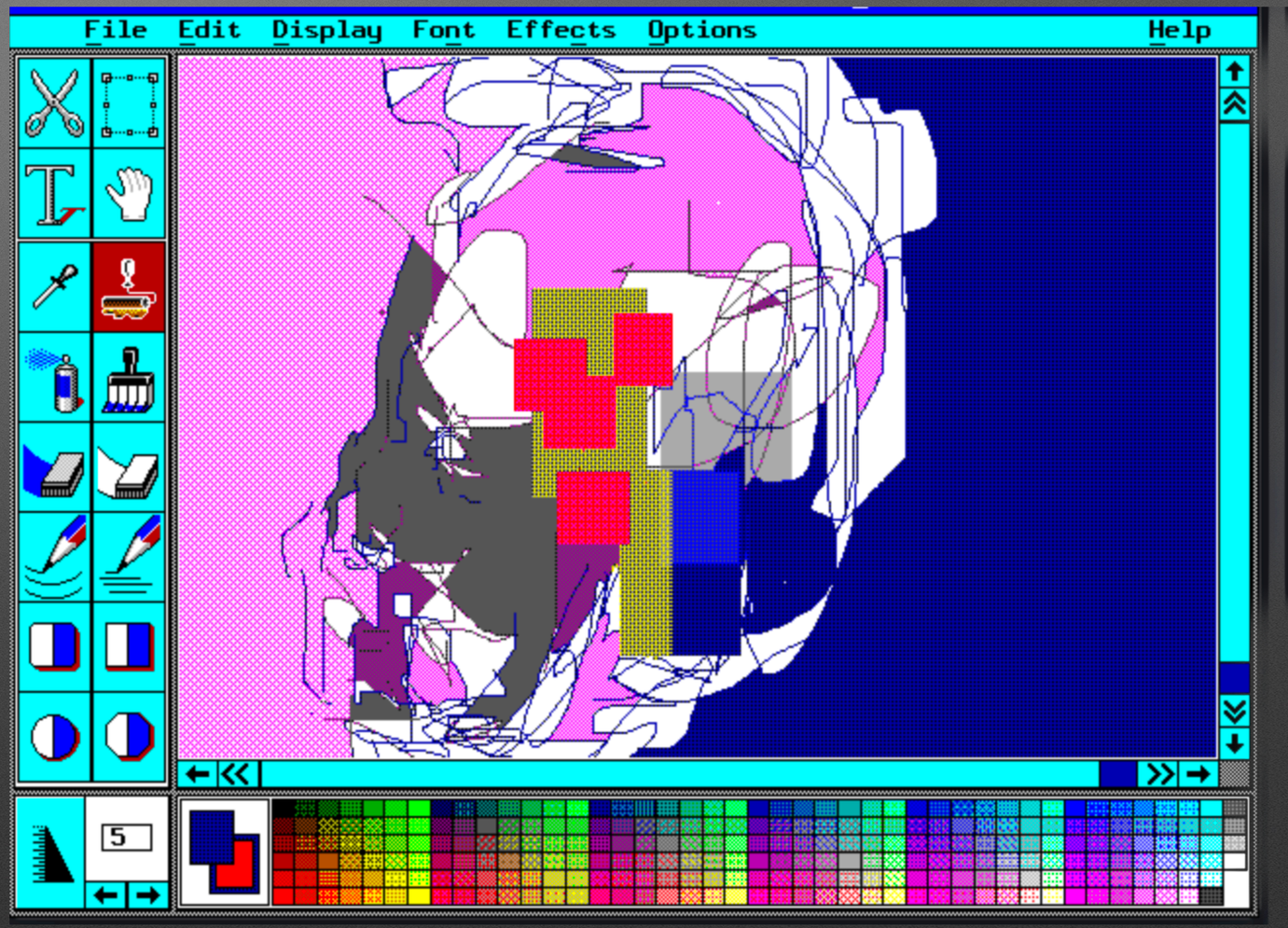
The views expressed in this presentation are my own and do not reflect the opinions of my past, present or future employers

Viewer Discretion is advised.

A Whirlwind Tour of MS-DOS

The DOS Kernel

- OS v1.0 debuted in 1981, v6.22 1994
- Some features of MS-DOS include:
  - MS-DOS operates in 16-bit real mode
  - Provides device-independent device access to computer resources, using the key programming interface of MS-DOS: ***system functions***
  - Single-task operating system [only one program runs at a time]**
    **TSRs are a partial workaround to the limitations of a single-task OS

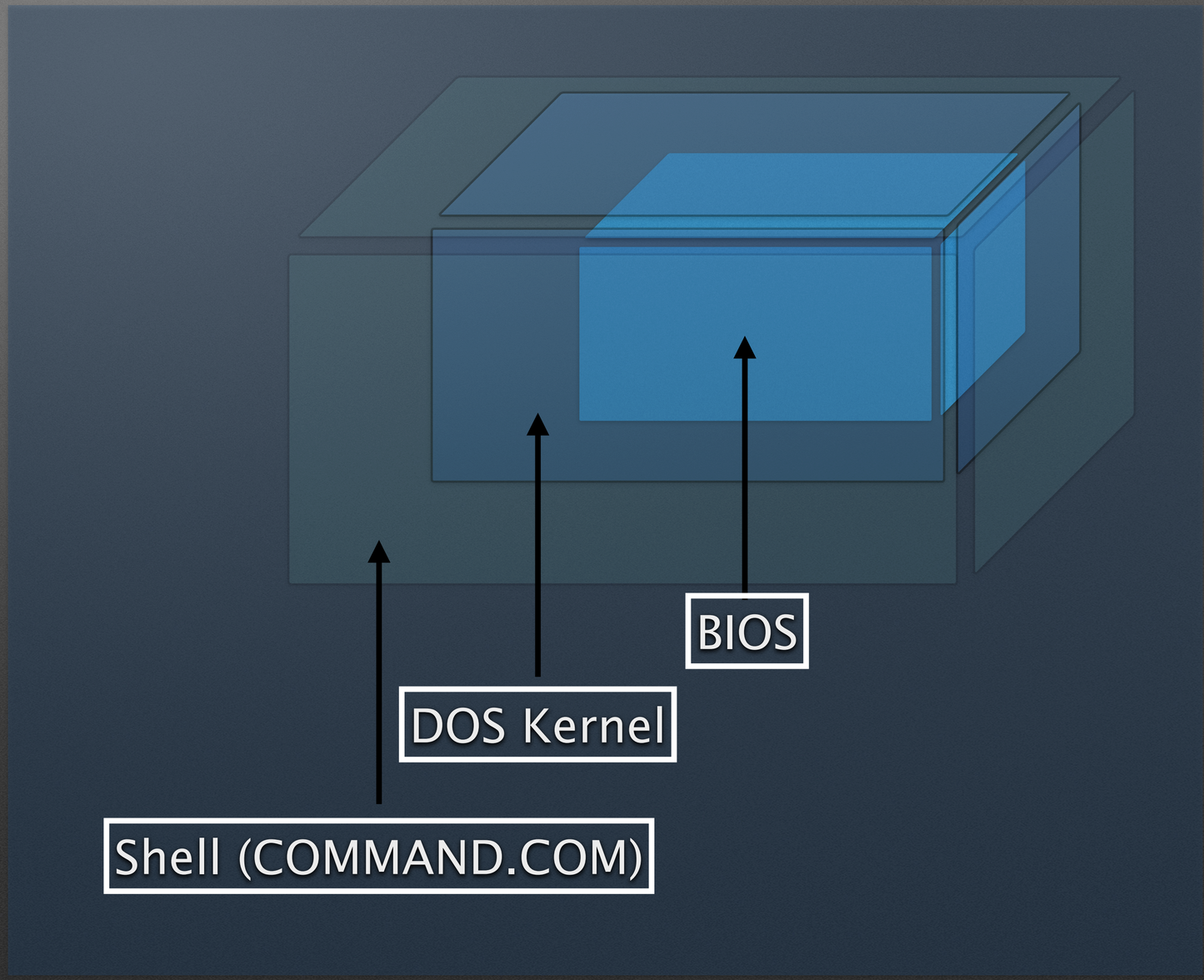


MS Paintbrush, you don't look a day over 1989 honey xoxo

"Microsoft MS-DOS Programmer's Reference," Microsoft Corporation, 2nd ed.: version 6.0., Microsoft Press, 1993

The DOS Kernel

- The MS-DOS operating system is divided into roughly three layers:
- 1. The BIOS (Basic Input/Output System)
- 2. The DOS Kernel
- 3. The command processor (shell) — COMMAND.COM

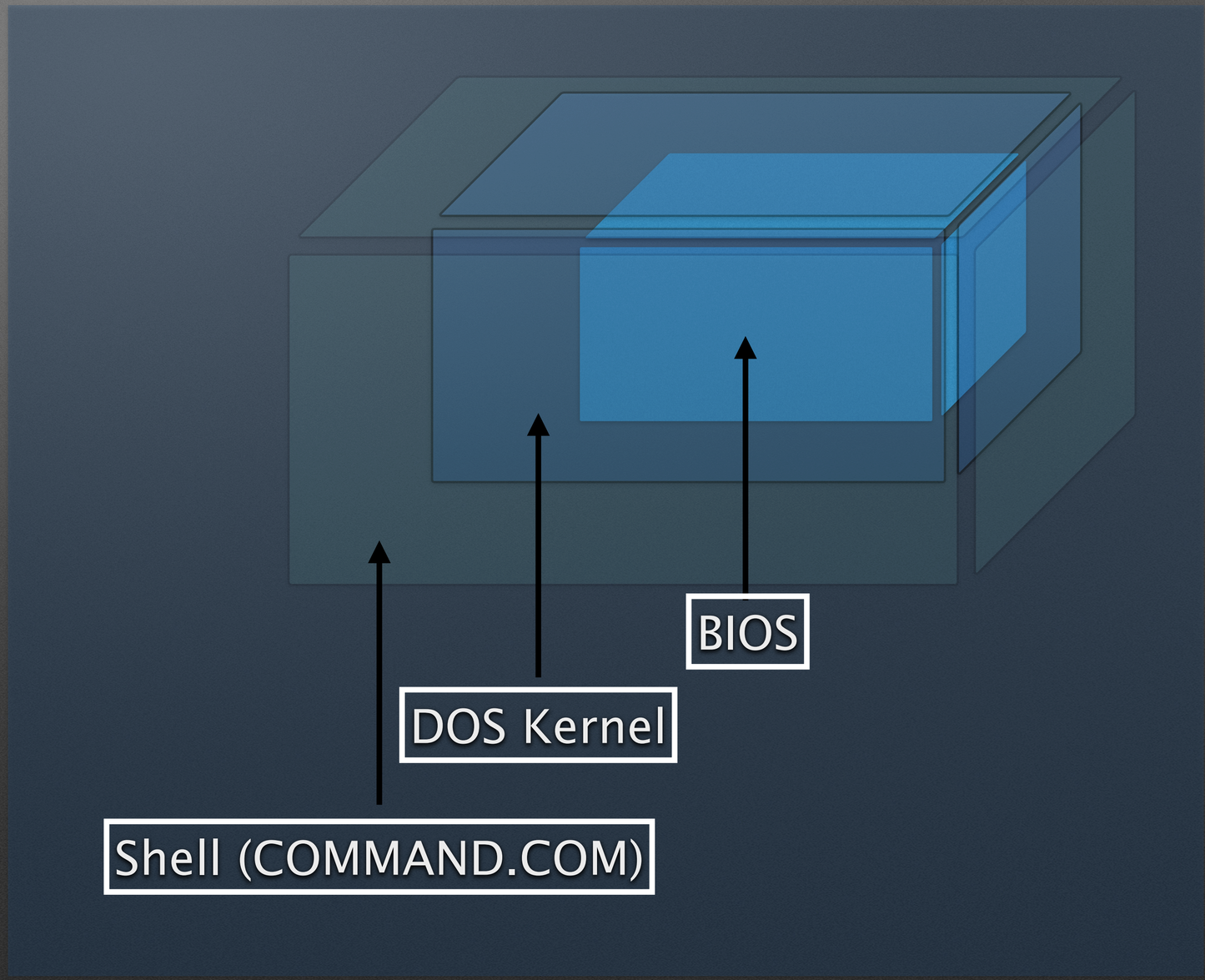


"Advanced MS-DOS Programming: Section 1 - Programming for MS-DOS," Ray Duncan, Microsoft Press, 1986

The DOS Kernel

- The DOS kernel provides ***system functions*** that allow a user to perform actions with provided collection of hardware-independent services
- These system functions include:
  - Memory management
  - Spawning programs
  - Character device I/O
  - File management
- Programs in MS-DOS interact with these system functions by loading registers with function-specific values + transferring control using ***software interrupts***

"Advanced MS-DOS Programming: Section 1 - Programming for MS-DOS," Ray Duncan, Microsoft Press, 1986



A Whirlwind Tour of MS-DOS: Notable Interrupts for Malware

Notable Interrupts for MS-DOS Malware

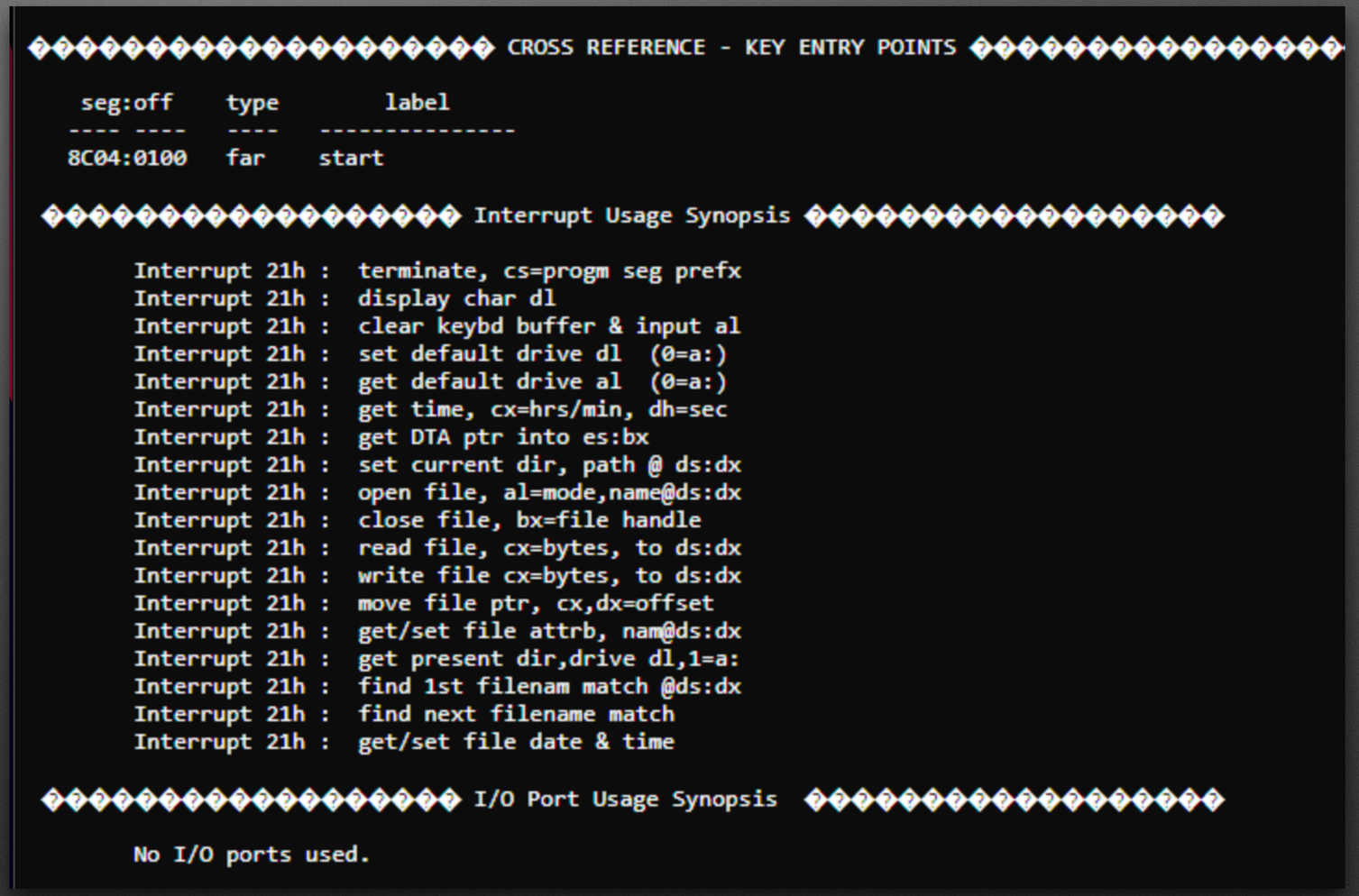
- System Interrupts (ROM BIOS):
  - Int 10h: Video services
  - ***Int 13h: Disk services***
  - Int 16h: Keyboard services
- MS-DOS Interrupts:
  - ***Int 21h - MS-DOS System Functions***
  - Int 25h - Absolute Disk Read
  - Int 26h - Absolute Disk Write

ROM Bios Interrupts are 05h, and 10h-1Fh

MS-DOS Reserved Interrupts 20h-3Fh

Notable Interrupts for MS-DOS Malware

- RTFMSDOSS (RTF MS-DOS Source)
- Because it’s usually beautifully + succinctly documented by the virus authors themselves
- Thanks x a mill R. Burger (this is a beautiful asm file)



```
CROSS REFERENCE - KEY ENTRY POINTS

   seg:off    type        label
  ---- ----   ----   --------------
  8C04:0100   far    start

Interrupt Usage Synopsis

      Interrupt 21h :  terminate, cs=progm seg prefx
      Interrupt 21h :  display char dl
      Interrupt 21h :  clear keybd buffer & input al
      Interrupt 21h :  set default drive dl  (0=a:)
      Interrupt 21h :  get default drive al  (0=a:)
      Interrupt 21h :  get time, cx=hrs/min, dh=sec
      Interrupt 21h :  get DTA ptr into es:bx
      Interrupt 21h :  set current dir, path @ ds:dx
      Interrupt 21h :  open file, al=mode,name@ds:dx
      Interrupt 21h :  close file, bx=file handle
      Interrupt 21h :  read file, cx=bytes, to ds:dx
      Interrupt 21h :  write file cx=bytes, to ds:dx
      Interrupt 21h :  move file ptr, cx,dx=offset
      Interrupt 21h :  get/set file attrb, nam@ds:dx
      Interrupt 21h :  get present dir,drive dl,1=a:
      Interrupt 21h :  find 1st filenam match @ds:dx
      Interrupt 21h :  find next filename match
      Interrupt 21h :  get/set file date & time

I/O Port Usage Synopsis

      No I/O ports used.
```

“Virdem” by R. Burger, 1989

A Whirlwind Tour of MS-DOS COM Programs

COM Programs

- .COM programs fit the TINY memory model of Intel 8086 ISA
- Must always have an origin of 100h [This is the length of the Program Segment Prefix or PSP]
- All segment registers contain the same value — code and data are mixed together
- No header, no identifying information, no relocation information
- No parents, no rules! (Not quite, but almost)

```
.286
.MODEL TINY

; *:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*:*
;       Hooks BIOS interrupts to draw pretty pictures to terminal screen
;       Intro to COM Programs for HUSHCON Seattle 2022 Presentation
;       MTV-Reboot My Super Sweet 16-Bit Malware:
;       ~*MS-DOS Edition*~
; **********************************************************************************************

.CODE
       org     100h

_start  PROC    NEAR
        mov     ax, 0B800h
        mov     es,ax
        mov     di,0h
        mov     cx,0h

sweet_n_init:
        xor     di,di

sweet_n_setup:
        mov     al,es:[di]
        mov     ax,di
        mov     es:[di],al
        jmp     sweet_n

sweet_n_right:
        mov     cx,50h
        mov     al,es:[di]
        add     ax,di
        mov     es:[di],al
        rep     stosw
```

COM Programs

- Max size of .COM program:
  65536 bytes - length PSP (256 bytes) - word of stack (2 bytes) = 65278 bytes (~63kB)

- .COM resides in memory as an absolute memory image

  - resides (is loaded into) a single segment of memory [a segment = 64k]

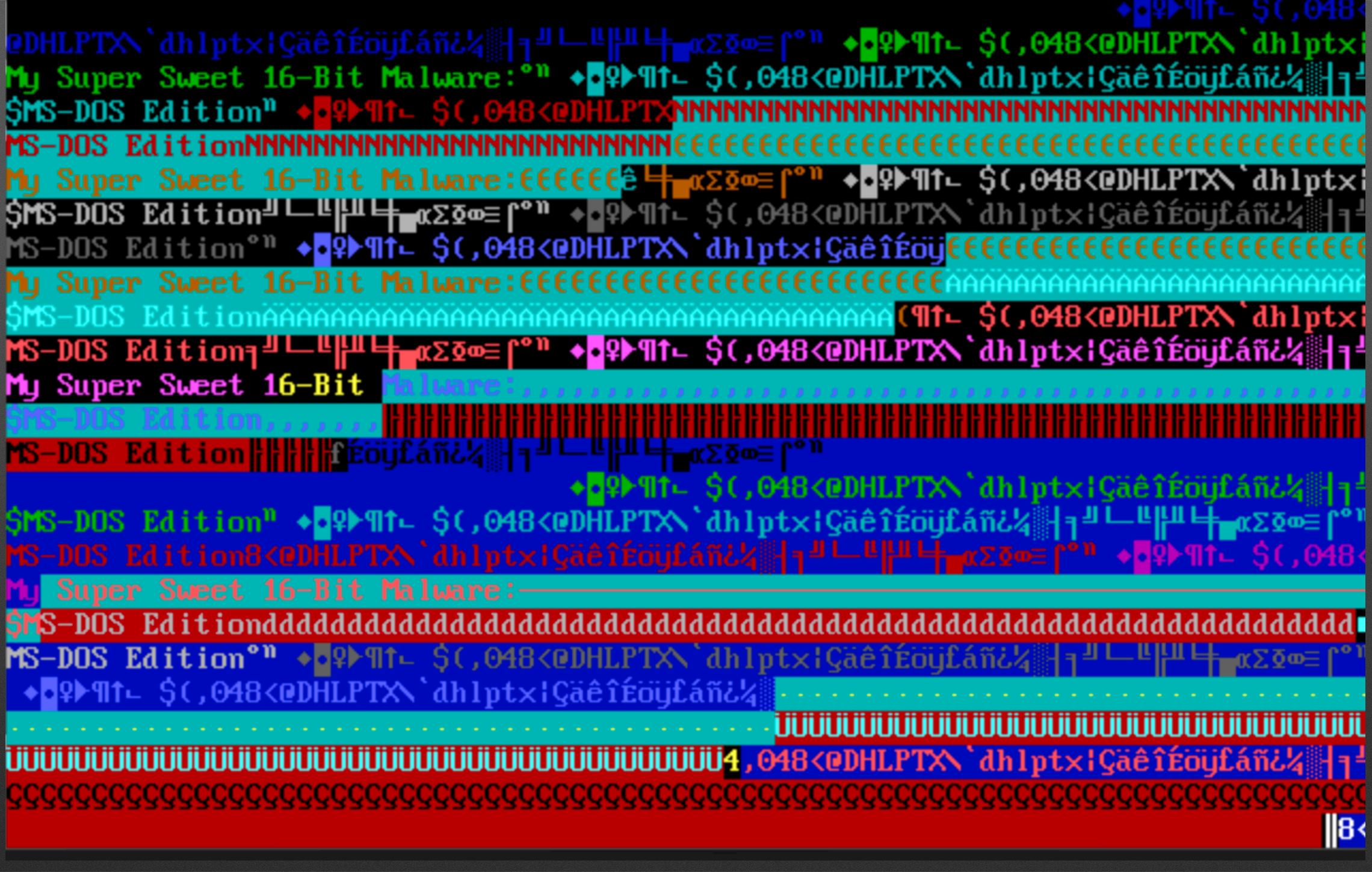
***

Uses segmented addressing scheme of 16-bit architecture (again we're running in 16-bit real mode, but accessing addresses in a range of a 20-bit address space)

[segment]:[offset]

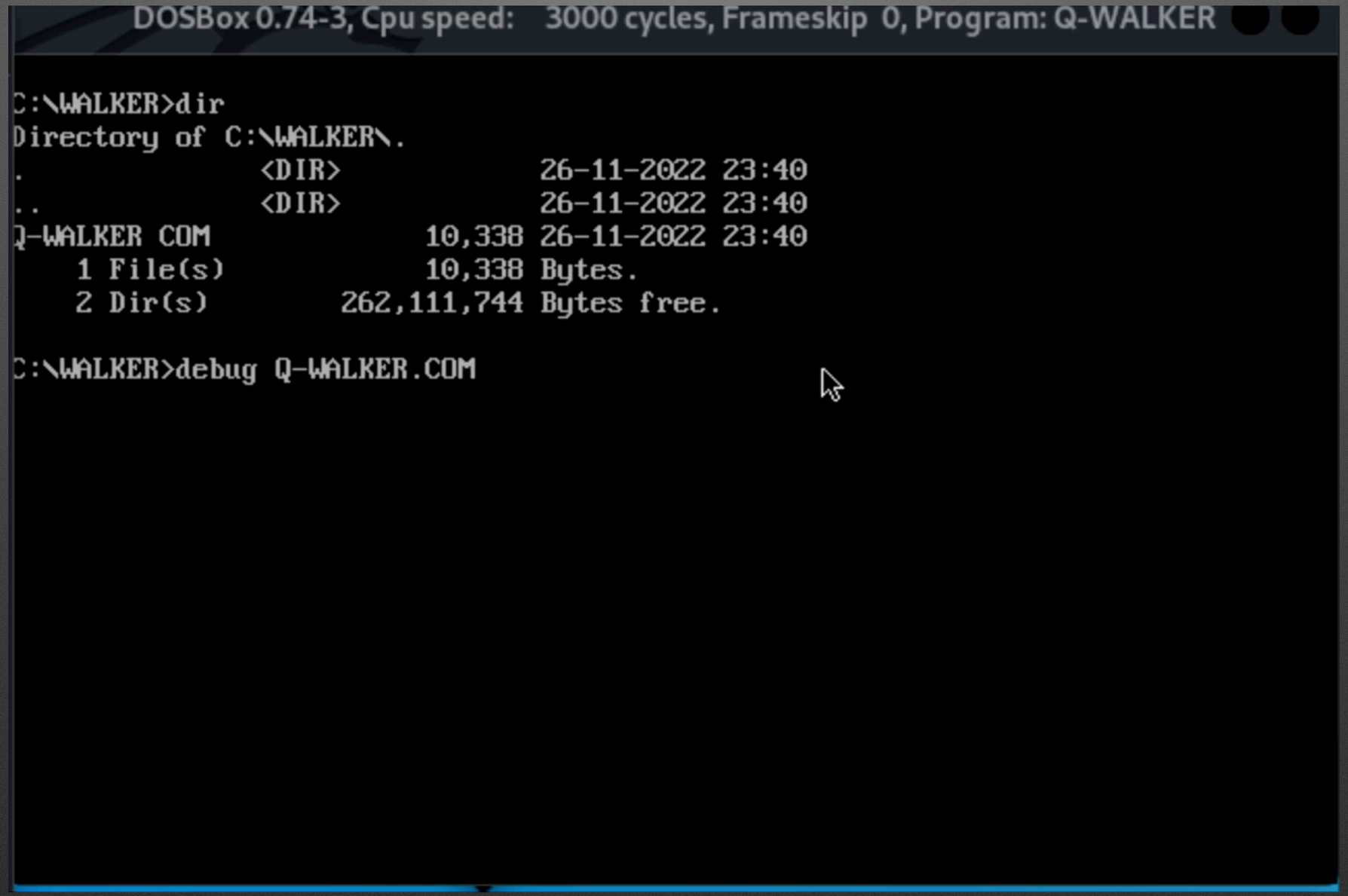
```
sweet_n_intro:
        mov     ah,40h
        mov     bx,1
        mov     cx,b_len
        mov     dx,offset b_msg
        int     21h

sweet_n_intro_2:
        mov     ah,40h
        mov     bx,1
        mov     cx,c_len
        mov     dx,offset c_msg
        int     21h
        jmp     sweet_n


sweet_n:
        mov     al,es:[di]
        add     ax,di
        mov     es:[di],al
        stosw

        mov     ah,0h
        int     16h
        ;; check if keypress is capital 'M' key
        cmp     al,50h
        je      sweet_n_intro

        cmp     al,018h
        jnz     sweet_n_setup
```

```
 sweet_n_epilogue:
         ;;end-program interrupt
         mov     ax,4C00h
         int     21h



 _start  ENDP

 b_msg   db    'My Super Sweet 16-Bit Malware:',0Dh,0Ah
 c_msg   db    '*~MS-DOS Edition~*',0Dh,0Ah
 ;message to display

 b_len   equ   $-b_msg
 c_len   equ   $-c_msg
```


My Super Sweet 16-Bit Malware:
MS-DOS Edition

Greatest Hits of MS-DOS Malware
or
“not just another pretty MBR bootkit”

WALKER


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: Q-WALKER
C:\WALKER>dir
Directory of C:\WALKER\.
. <DIR> 26-11-2022 23:40
.. <DIR> 26-11-2022 23:40
Q-WALKER COM 10,338 26-11-2022 23:40
1 File(s) 10,338 Bytes.
2 Dir(s) 262,111,744 Bytes free.
C:\WALKER>debug Q-WALKER.COM

MS-DOS Techniques in the MITRE ATT&CK Framework [non-extensive but wow this list looks so boring you wouldn’t know it!]

- Data Manipulation [T1565] https://attack.mitre.org/techniques/T1565/
- Replication through Removable Media https://attack.mitre.org/techniques/T1091/
- ***Masquerading*** https://attack.mitre.org/techniques/T1036/004/
- Masquerading: Match Legitimate Name or Location [T1036:005] https://attack.mitre.org/techniques/T1036/005/
- Masquerading: Masquerade Task or Service [T1036:004] https://attack.mitre.org/techniques/T1036/004/
- Data Obfuscation [T1001] https://attack.mitre.org/techniques/T1001/
- System Services [T1569] https://attack.mitre.org/techniques/T1569/
- Direct Volume Access [T1006] https://attack.mitre.org/techniques/T1006/
- File and Directory Discovery [T1083] https://attack.mitre.org/techniques/T1083/
- Boot or Logon Autostart Execution [T1547] https://attack.mitre.org/techniques/T1547/
- Defacement [T1491] https://attack.mitre.org/techniques/T1491/
- Pre-OS Boot: Bootkit

MS-DOS Malware Techniques

Classic Malware
Stealth
+
Persistence

Level 10 SAVAGE
Destruction of the MBR

Exquisite Graphical Rendering/
Data Manipulation
using system functions

Sources

- vx-underground — MS-DOS Malware collection
- “Internet Archive — Malware Museum,” Mikko Hypponen, https://archive.org/details/malwaremuseum
- danooct1 YouTube channel: https://www.youtube.com/@danooct1
  Specifically their “MS-DOS malware” playlist: https://youtube.com/playlist?list=PLi_KYBWS_E71ObQ8QpGj5zIDXHREbdWaM

Greatest Hits of MS-DOS Malware:

~~The Clash~~ CRASH.COM

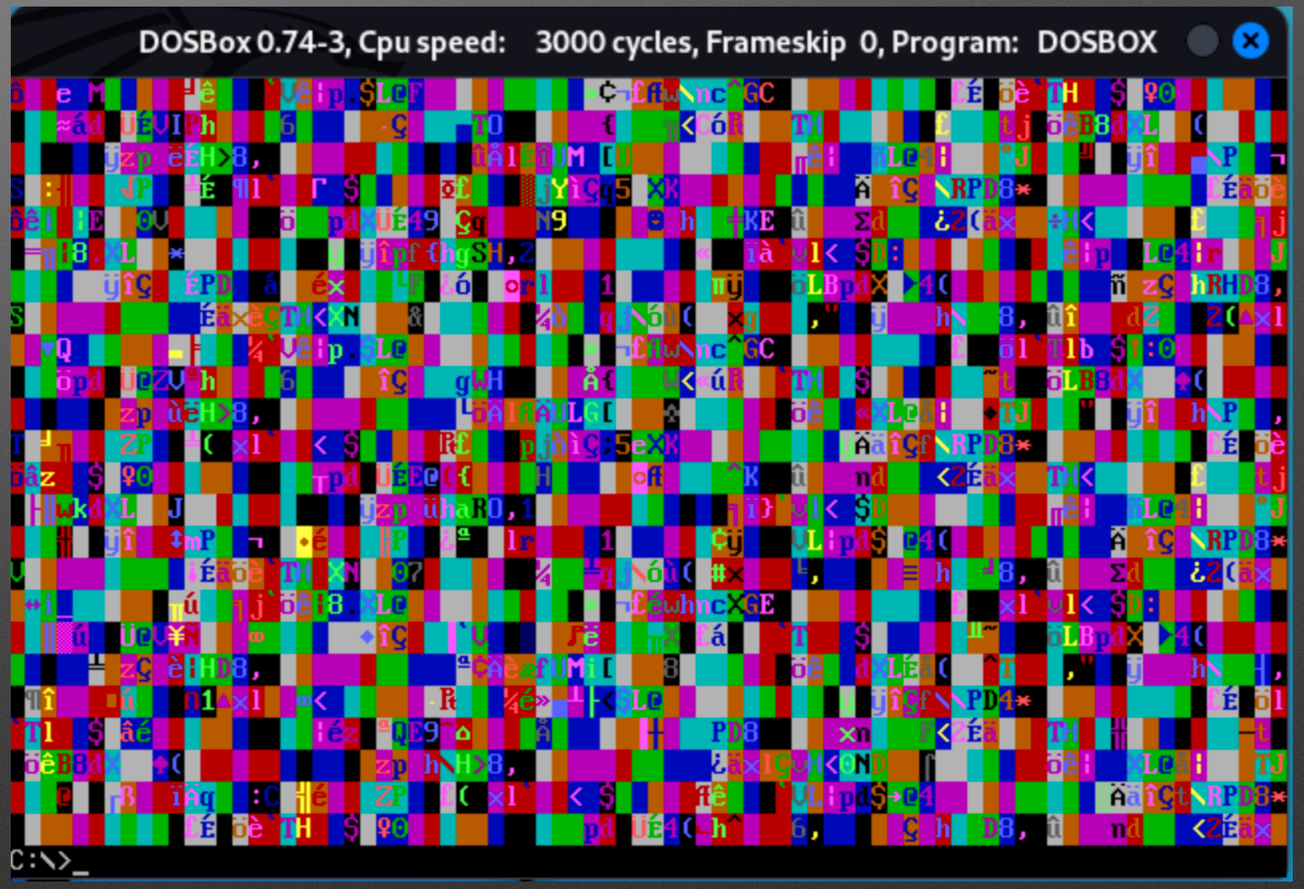
CRASH.COM

Infinite loop of pretty animation
Direct write to the VGA buffer makes the computer unusable and forces a user to reboot to use their machine

Copies the payload [~*pretty animation on infinite loop*~] to target files on the machine

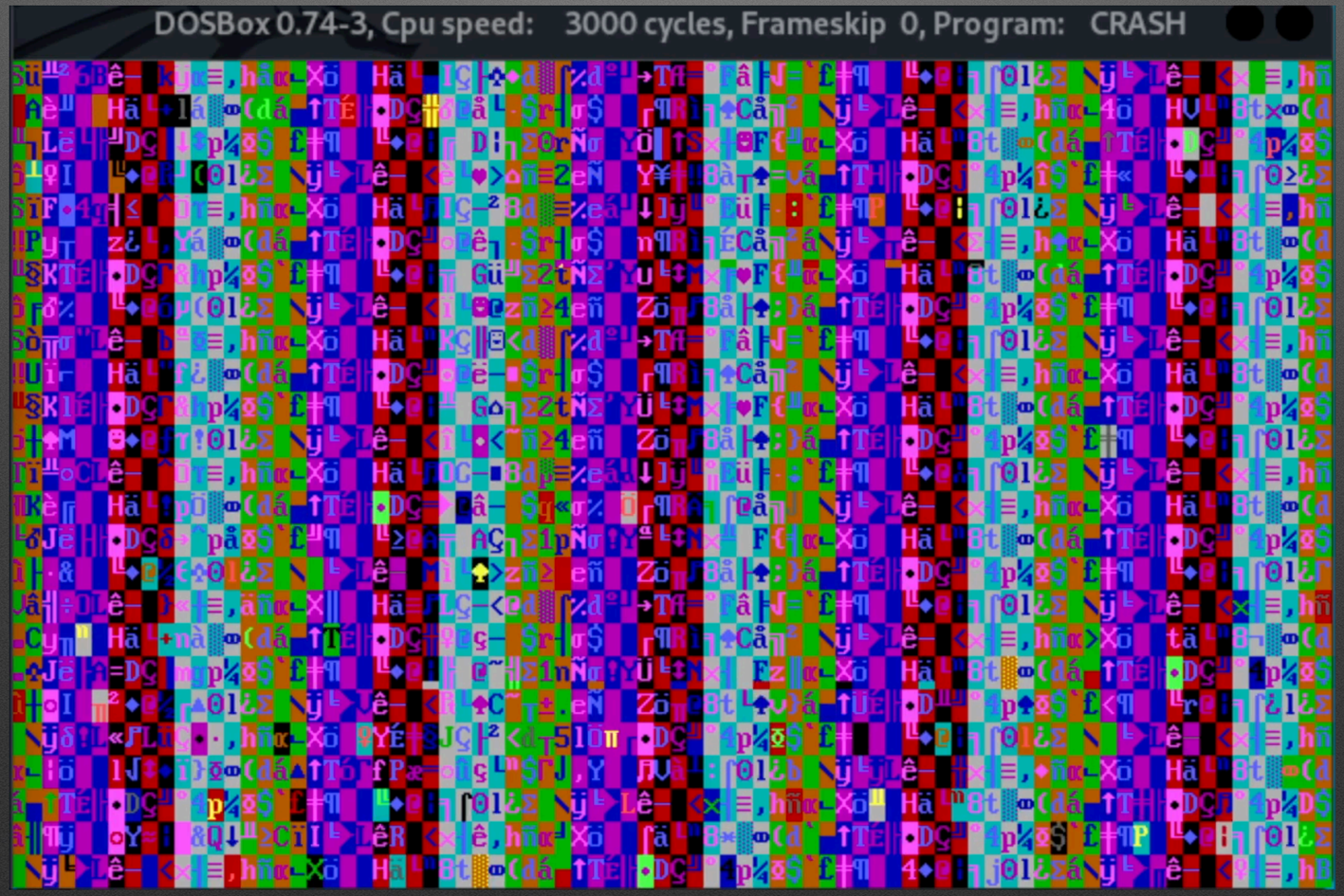
Less destructive ** than other viruses of the time that used this same VGA buffer technique

** I do not have access to the original source, only a modified version, so my analysis is based off of that version; if you have the original, please send it my way <3 <3



CRASH.COM

WARNING: Mild Flashing Lights [appx. 6 seconds]



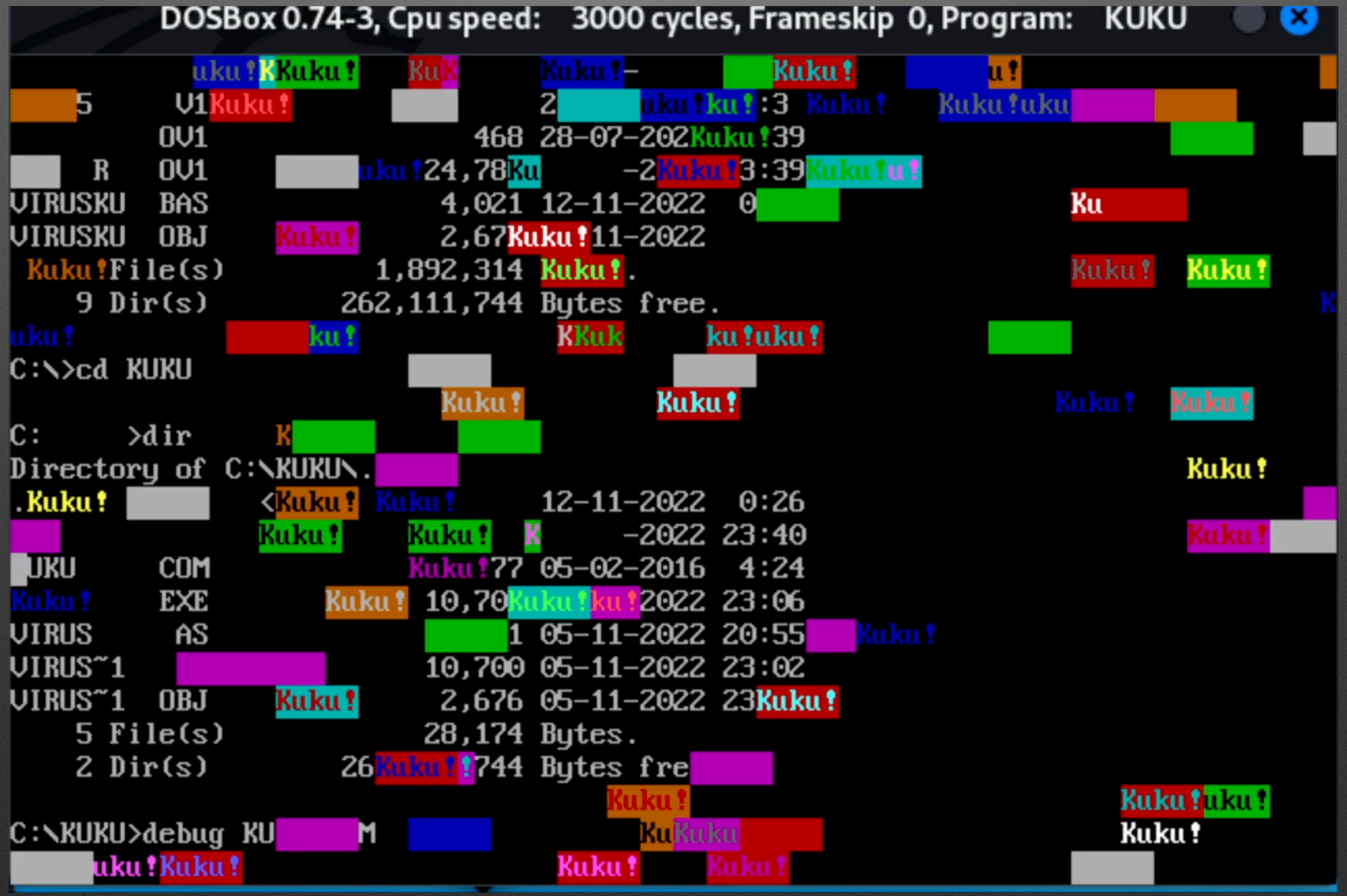
Greatest Hits of MS-DOS Malware

~~The Kooks~~ KUKU.COM

KUKU

- Searches the filesystem for .exe and .com files, overwrites them with the KUKU.COM virus payload
- When executed, displays this obscene sequence of colorful boxes with the phrase “Kuku!” to the command prompt
- Fun (?) fact: kuku (kyky) means “peekaboo” in Russian. The virus is searching for target files to infect and displaying a message “peekaboo, I see you!” every time it finds one.

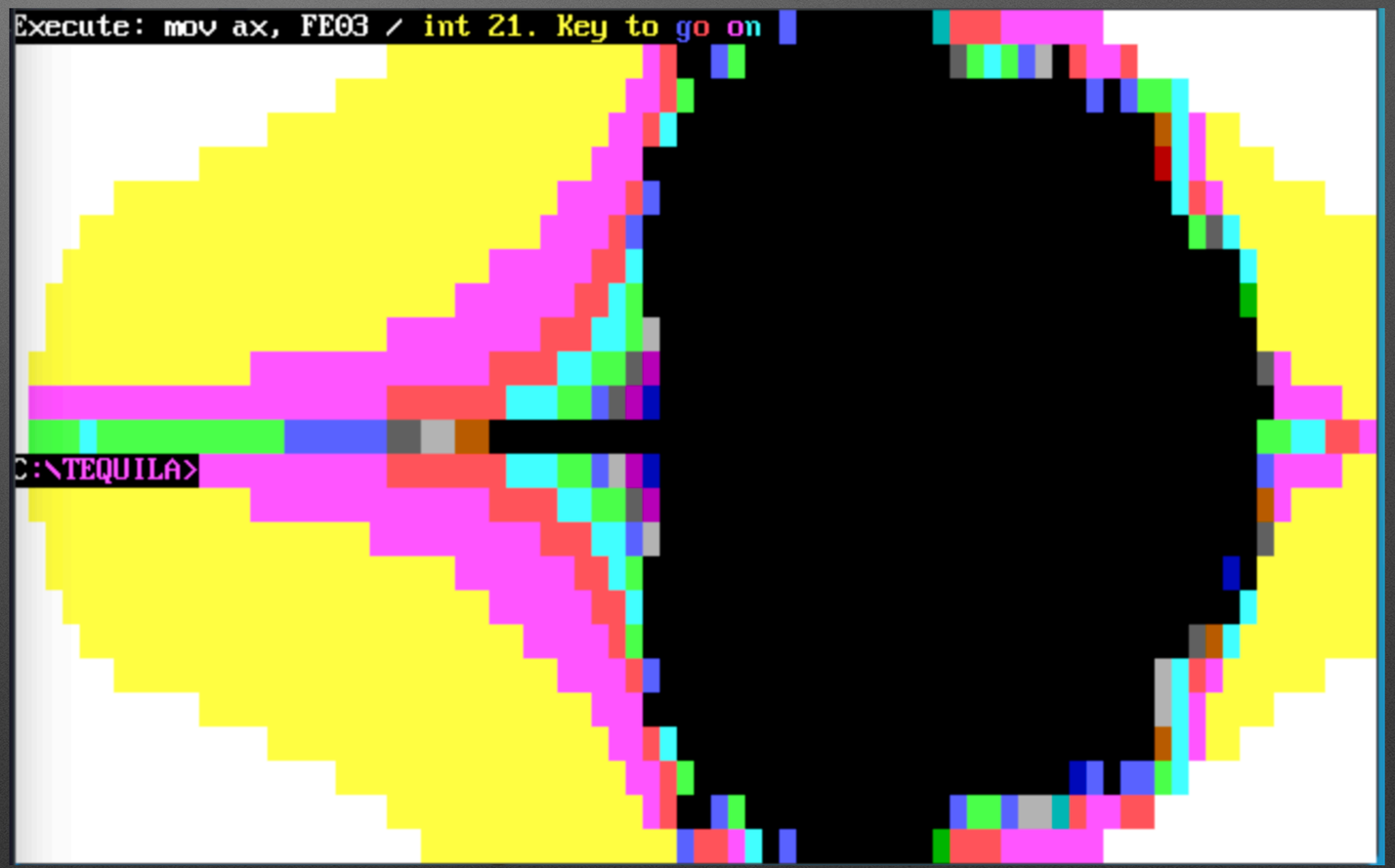
KUKU


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: KUKU
C:\>cd KUKU
C:\KUKU>debug KU

Greatest Hits of MS-DOS Malware

TEQUILA

- Fractal animation
- Savage - infects MBR partition table and installs interrupt handlers to run as a TSR
- What's with the "Mov ax FE03 / INT 21" instruction??
- Multi-part payload — requires user interaction to reveal …



TEQUILA

```
DB 00DH, 00AH, 00DH, 00AH
DB "Welcome to T.TEQUILA's latest production.", 00DH, 00AH
DB "Contact T.TEQUILA/P.o.Box 543/6312 St'hausen/"
DB "Switzerland.", 00DH, 00AH
DB "Loving thoughts to L.I.N.D.A", 00DH, 00AH, 00DH, 00AH
DB "BEER and TEQUILA forever !", 00DH, 00AH, 00DH, 00AH
DB "$"

DB "Execute: mov ax, FE03 / int 21. Key to go on!"
```

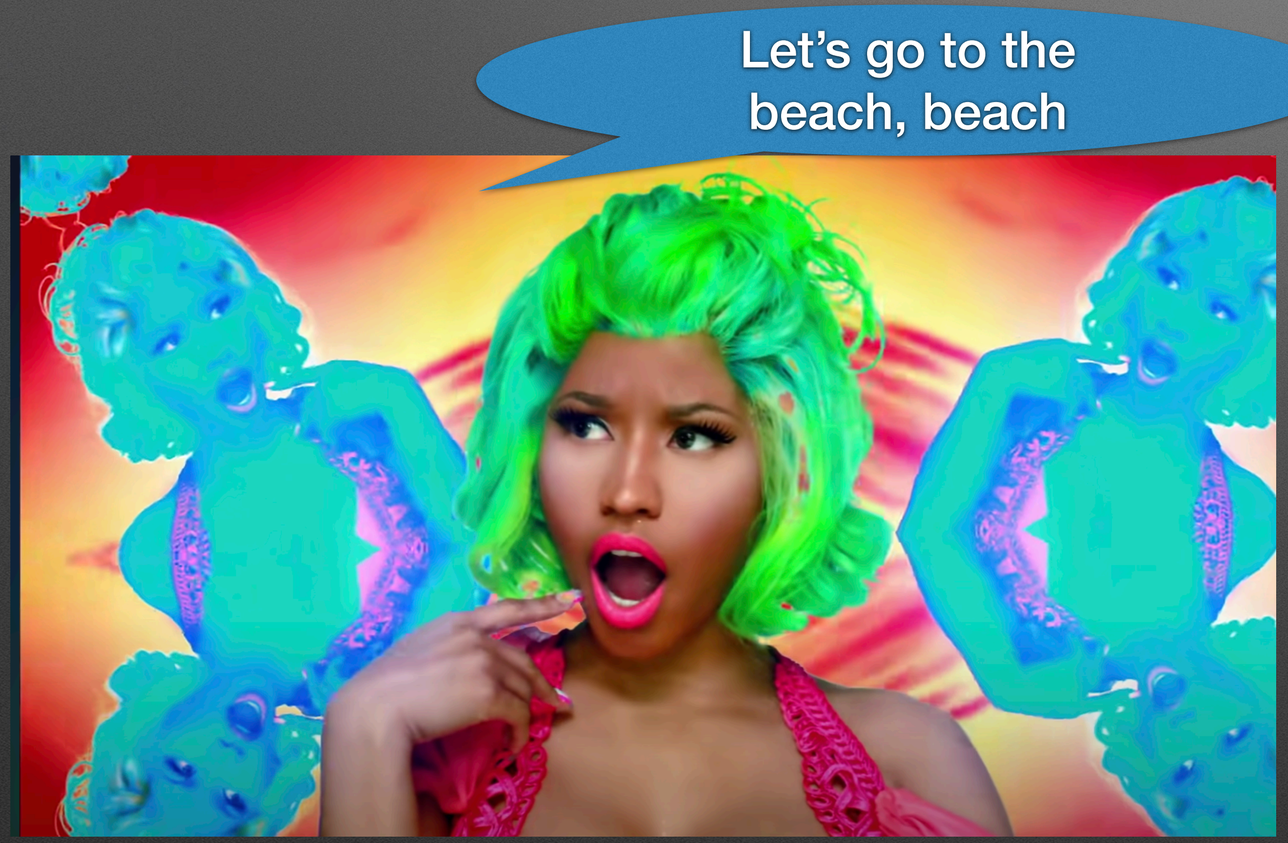
Greatest Hits of MS-DOS Malware

~~Yellow Submarine~~ MARINE.COM

MARINE

- LEVEL 10 Savage
- Shreds the MBR while a little boat animation plays
  - Specifically it encrypts the drive
- “Всё na mope” in Russian means “everyone to the ocean” but the vibe is basically like…

MARINE


Let’s go to the
beach, beach


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: MARINE
ВСЕ НА МОРЕ !!!

Все хорошо

Nothing bad happening to the drive rn

MARINE

Why Study an EOL OS?

Malware — same as it ever was…

"Once in a Lifetime," The Talking Heads, 1980

Why Study an EOL OS?

- Foundational techniques of malware masters of the ‘80s inform malware techniques up to the present day
- For one, legacy BIOS interrupts featured prominently in malware like bootkits through much later versions of Windows [specifically through Windows 7 before the switch to UEFI from the legacy boot process]
- Relevant case study: NotPetya
  - Prominent use of the INT 13h BIOS interrupt to call disk services functions
- And while the switch to UEFI firmware from the legacy boot process (meaning the use of legacy BIOS interrupts) effectively changed the landscape for Windows bootkits, there may still be other areas of the kernel, using the same ol’ INT 10h and INT 16h, and maybe, if an attacker were really really lucky that notorious, devious wicked INT 21h.
- In any case, studying these samples, offers new insights into malware that attacked versions of Windows, using the same legacy BIOS interrupt system, up through Windows 7. And if you work in this space, hit me up after this talk!

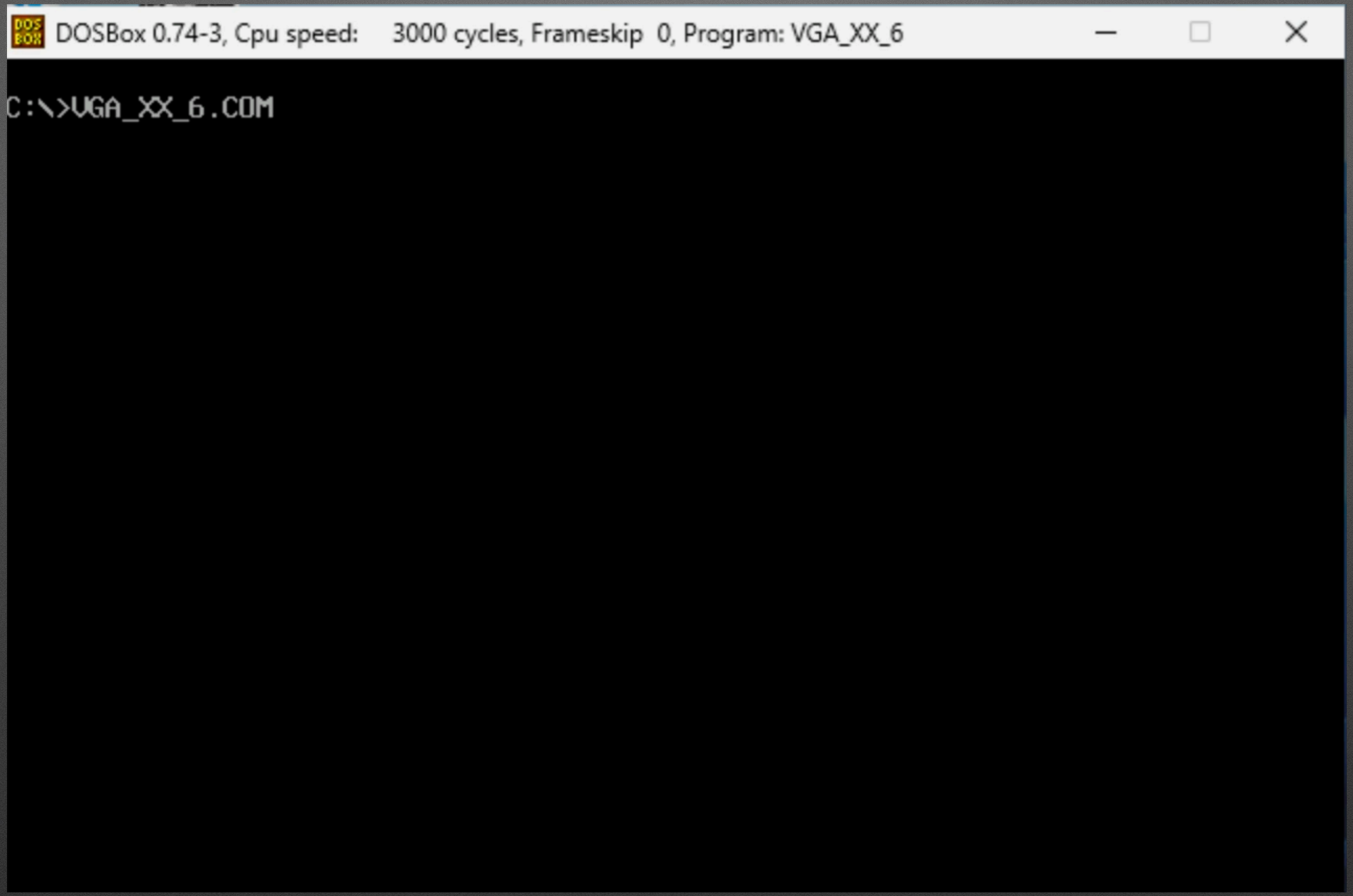
“Rootkits and Bootkits: Reversing Modern Malware and Next Generation Threats,” Alex Matrosov, Eugene Rodionov, and Sergey Bratus

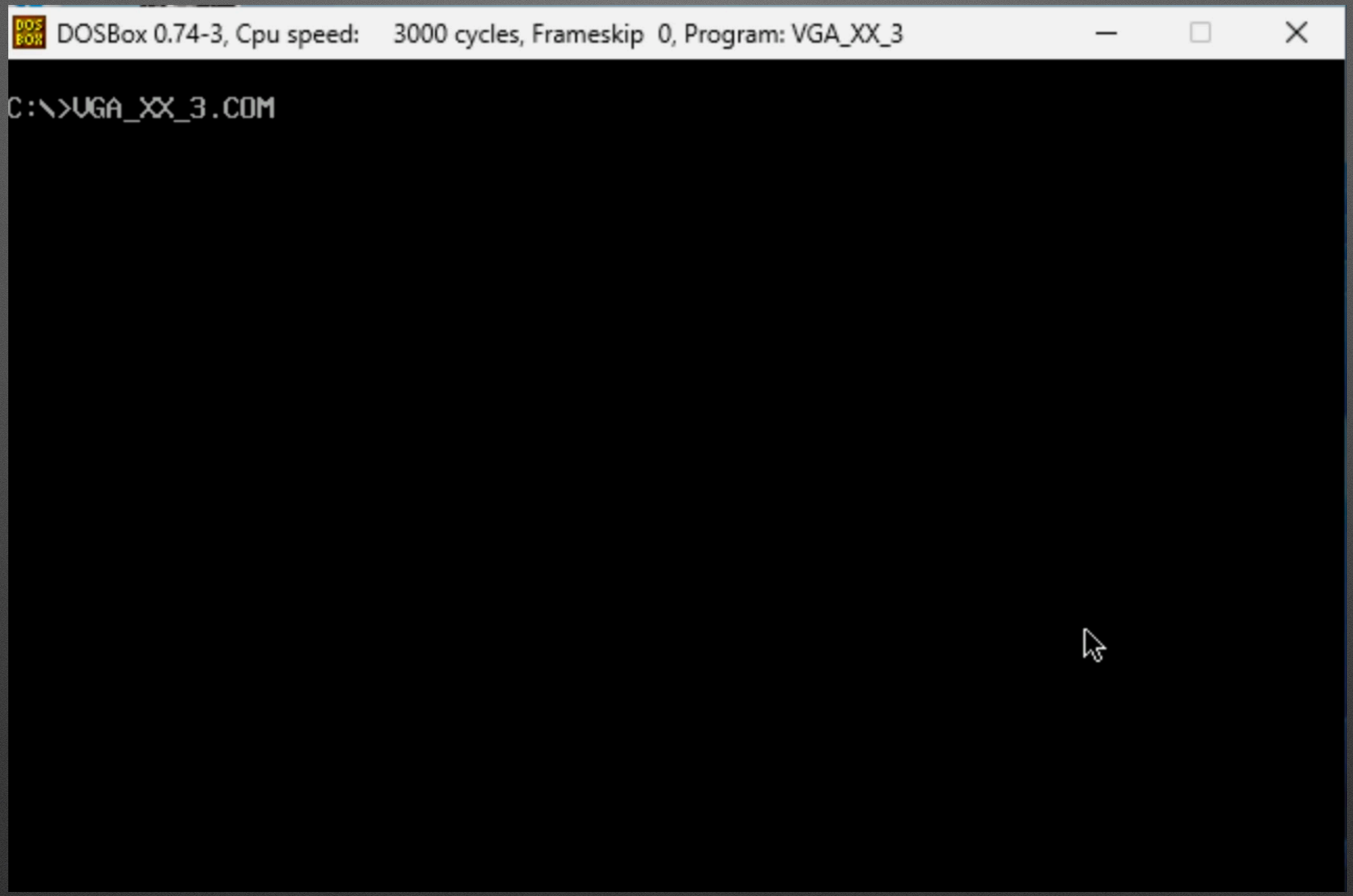
Where to now?

- I have a repo on GitHub with some sample COM programs, and will be adding resources for setting up a reversing lab: (Work in progress, expect commits with new material in the coming weeks):
  https://github.com/nikaroxanne/supersweet16bit-m4lw4r3

- I'm also working on some plugins for r2 and IDA for reversing 16-bit binaries (specifically malicious COM files); contact me on one of the socials (on in person at Hushcon) if you would like to be involved in ~*test driving*~ those

- I made you a playlist for when you're reversing 16-bit malware:
  https://open.spotify.com/playlist/7KKMdu8JaLEoLV66uZqKG0?si=bf876ac6c71f4ecf
  [If you are adamantly opposed to Spotify as a platform, then provide me with your medium of choice and I will burn you a copy. Bonus points if that medium of choice is a FAT stack of floppy disks (someone better appreciate this joke I stg)]


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: VGA_XX_6
C:\>VGA_XX_6.COM


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: VGA_XX_3
C:\>VGA_XX_3.COM

References

“Advanced MS-DOS Programming,” Ray Duncan, Microsoft Press, 1986

“Microsoft MS-DOS Programmer’s Reference,” Microsoft Corporation, 2nd ed.: version 6.0., Microsoft Press, 1993

“A Look Back at Memory Models in 16-bit MS-DOS,” Raymond Chen, Microsoft Blogs, July 28, 2020, https://devblogs.microsoft.com/oldnewthing/20200728-00/?p=104012

“On Memory Allocations Larger Than 64KB on 16-Bit Windows,” Raymond Chen, Microsoft Blogs, https://devblogs.microsoft.com/oldnewthing/20171113-00/?p=97386

“Retired Malware Samples, Everything Old is New Again,” Lenny Zeltser, August 1, 2018. https://zeltser.com/retired-malware-samples-retrospective/

“Rootkits and Bootkits: Reversing Modern Malware and Next Generation Threats,” Alex Matrosov, Eugene Rodionov, and Sergey Bratus