

Defining a Computer Virus

- A virus is an entity that uses the resources of the host to spread and reproduce itself, usually without informed operator action.
- A virus cannot execute on its own.
- Strong viruses use normal computer operations to achieve the virus design goals.
- *There is no single characteristic that can be used to identify a previously unknown virus program.*
- Consequently, there is some academic disagreement as to just how many viruses have been released, what variants define different strains.



Virus Structure

- **Infection:** The infection mechanism may be defined as the way or ways in which the virus spreads.
- **Payload:** The payload mechanism is defined as what (if anything) the virus does *in addition* to replicating.
- **Trigger:** The trigger mechanism is defined as the routine that decides what time to deliver the payload if there is a payload.



Virus Damage

- **Deliberate damage** inflicted by the virus payload mechanism, if it exists, such as the trashing or intentional corruption of files.
- **Accidental damage** caused when the virus attempts to install itself on the victim system (the newly infected host), such as corruption of system areas preventing the victim system from booting.
- **Incidental damage** that may not be obvious but is nevertheless inherent in the fact of infection. Nearly all viruses entail damage in this category, since their presence involves loss of performance due to theft of memory, disk space, clock cycles, system modifications or combination of these,



Some Social Impacts

- **Scapegoating of virus victims**
- **Secondary damage to systems caused by inappropriate responses to a perceived virus threat (ex. low-level formatting of a hard disk to eradicate a macro virus.)**
- **Legal or quasi-legal issues such as failure to comply with data-protection legislation and policies.**
- **Inappropriate security responses**
 - reformatting
 - passwords
 - change in business models



A Few Examples of Virus Damage

- The disappearance of Word menu options relating to the presence of macros.
- Encryption or displacement of system areas, such as the Master Boot Record.
- Manipulation of the Windows Registry
- Trashing or corruption of legitimate macros as part of the installation of a macro virus.



Latency

- **Unexecuted viruses are latent or dormant**
 - ex. mailbox full of unread, infected mail
 - ex. PC-specific virus residing on a Mac or a UNIX server.
- **“Heterogeneous virus transmission.”**



Some Useful Terms

- **Intendeds:** reproductive mechanism never triggers, or if triggered, code never attaches to host.
 - ex. virus intended to execute on Sundays and uses DOS system call Get Date. Virus waits for Get Date to return “7” but Get Date only returns values between “0..6.”
- **Corruptions:** may be caused by system transfers, incomplete “cleansing” and poorly maintained virus collections.
 - Antivirus programs often detect corrupted non-viral programs simply to avoid being penalized by incompetent testers and reviewers.



Virus Design Considerations

- **Polymorphic Viruses** – change structure in attempts to avoid detection
- **Non-Resident (direct action) versus Memory-Resident viruses.**
 - Hybrids
 - Macros
- **Payload versus reproduction**
- **Damage**
 - In general, a virus can do anything any other software can do
- **Boot Sector**



Attaching viral code to an existing program

- Overwrite existing program code (overwriting viruses)
- Add code to the beginning of the program (prependers)
- Add code to the end of the program (appenders)
- Insert viral code into the command chain so that it is run when the legitimate code is executed (parasitic viruses or file infectors)
 - Macro viruses are a special case of a file infector
- These methods are becoming less common as VBScript, AOL programs and MS Office macros continue to ease the task of virus writers.



Polymorphic Virus Techniques

- **Objective:** fool scanners, make signatures harder to identify
- **Methods**
 - **Encryption:** Start with a “random” number such as the value of seconds in system time then use that as a key to encrypt part of the payload.
 - **Arbitrary code relocation:** rearrange code after each infection.
- **Detection**
 - change detection
 - activity monitoring
 - detecting the mutating engine in kit-produced viruses
 - bankruptcy of scanners that cannot detect polymorphic viruses



Stealth Technology

- **Even if the virus is new or polymorphic, it still adds code to the infected program increasing size.**
 - If overwrites are carefully managed to avoid increasing size, new code will still fail a CRC.
 - For a review of CRC see http://www2.rad.com/networks/1994/err_con/crc.htm
- **Traps – a stealthy virus will try and intercept system calls to avoid detection**
 - Determine another program is trying to access the memory the virus is occupying so hang the system
 - Trap the attempt to read the disk information and return an image of the disk information before infection



Virus Symptoms

- **System software, applications, or utilities display unpredictable behavior.**
- **GPFs and similar conflicts and errors are encountered**
- **Parity and checksum errors are encountered**
- **Loss of performance**
 - **ex. loss of 32-bit access**
- **Loss of access to system areas may be observed, possibly entailing lost access to normally mounted volumes and subsequent unavailability of data and/or applications.**



Review of Boot Sequence

1. The user powers up the computers
2. Computer runs a power supply self-test
3. ROM BIOS code is executed
4. ROM BIOS performs a test of the central hardware
5. Computer runs a video test
6. Computer runs a memory test
7. On a cold boot, the full POST is run, skipped on a warm boot.
8. Computer tests for the partition boot record at the first sector of the default boot drive (specified in the BIOS).



Review of Boot Sequence (2)

9. The partition boot record is executed.
10. The computer initializes specified system files, or displays a message if these are not available (in DOS IO.SYS and MSDOS.SYS, in NT, NTLDR, NTDETECT.COM checks hardware and NTOSKRNL.EXE initializes the OS)
11. The base device drivers are initialized and device status is checked.
12. The computer reads configuration files (config.sys, system.dat, user.dat, as per OS).
13. The command shell (command.com for example) is loaded.
14. The shell's start-up command files (autoexec for example) are executed.



Virus Writing

- **Assembler versus High-Level Language**
- **For DOS-based attacks, a wide variety of file extensions are available.**
- **Consider:**
Program Screen_Virus;
const attack = 'Get a life'
begin
 writeln(attack);
end.
- **Turbo Pascal compiles this program in 1,920 bytes, MS Assembler takes 30 bytes.**
- **Much easier to access the boot sector or other low-level mechanisms via assembler.**



Tripartite Structure of a Virus (1)

- **Infection**

- **Begin**

- If (infectable_object_found)

- AND (object_not_already_infected)

- THEN (infect_object)

- **may entail**

- writing of a new section of code to the boot sector
 - addition of code to a program file
 - addition of macro code to MS Word NORMAL.DOT file
 - addition of code to a standard system program to intercept network services so as to send an infected file attachment to harvested email addresses



Tripartite Structure of a Virus (2)

- **Trigger**

Begin

IF (date_is_Friday_13th)

THEN (set_trigger_status_to_yes)

End

- **Payload**

Begin

IF (trigger_status_is_yes)

THEN (execute_payload)

End



Operating System Security

Trojan Horses

- **Does NOT self-replicate**
- **Free program made available to unsuspecting user**
 - Actually contains code to do harm
- **Place altered version of utility program on victim's computer**
 - trick user into running that program
 - la
 - /usr/mal/ls
- **Rootkits**
- **Remote Access Tools**
 - PCAnywhere
 - Laplink
 - Back Orifice

