Computer System Security

Assoc. Prof. Noha A. Hikal

Information Technology Dept.

Faculty of Computers and Information Sciences

Subject Code:IT424

4th level –IT&IS

Week 7 - 23 March 2020

Programs Security

Outlines:

1. Programs Flaws
2. Non-malicious codes
3. Malicious codes
4. Program development controls against malicious code and vulnerabilities: software engineering principles and practices
5. Controls to protect against program flaws in execution: operating system support and administrative controls

Program/Code Security Goals:

- To keep programs free from flaws

- To protect computing resources against programs that contain flaws

1- Programs Flaws

Bug, Error, Fault, and Failure

Bug in software is a term that can mean many different things depending on context. For example, it can be a mistake in interpreting a requirement or a syntax error in a piece of code.

Error is a human mistake in performing some software activity that may lead to a **fault** in a computer program.

A **fault** may cause a **failure** (which is a departure from the system's required behavior).

a fault is an inside view of the system, seen by the developers, whereas a failure is an outside view seen by the user.

Types of Flaws

- **Intentional**
  - Malicious
  - Non-malicious

- **Inadvertent (Unintentional)**
  - **Validation error**.

    (incomplete or inconsistent): permission checks
  - **Domain error**

    controlled access to data
  - **Inadequate identification and authentication.**

    basis for authorization
  - **Boundary condition violation**

    failure on first or last case
  - **Other exploitable logic errors**.

2- Non-malicious Program Flaws

Non-malicious Program Errors

Three classic error types that have enabled many recent security breaches.

1. Buffer overflow
2. Incomplete mediation
3. Time Of Check To Time To Use

Non-malicious Program Errors (cont.)

- **Buffer Overflows**

All program and data elements are in memory during execution, sharing space with the operating system, other code, and resident routines. Therefore, the effect of the overflows data depends on where it is go in the memory; It may affect the user data, user code, system data, or system code.

Example:

```
char sample[10];
for (i=0; i<=9; i++)
    sample[i] = 'A';
sample[10] = 'B'
```

Non-malicious Program Errors (cont.)

- **Incomplete Mediation**

http://www.things.com/order.asp?custID=101&part=555A&qy=20&price=10&ship=boat&shipcost=5&total=**205**

http://www.things.com/order.asp?custID=101&part=555A&qy=20&price=10&ship=boat&shipcost=5&total=**25**

Non-malicious Program Errors (cont.)

- **Time-of-Check to Time-To-Use Errors (TOCTTOU)**

A delay between the time the access was checked and the time the result of the check was used.

A change occurred, invalidating the result of the check.

Example: Changing the file name that checked for a deletion access.

3- Malicious Codes

Virus	is a malicious code that attaches itself to program that can replicate itself and to perform a malicious action to other non-malicious programs by modifying them.
Trojan horse	is a malicious code that, a program that overtly does one thing while covertly doing another malicious one
Logic bomb	is a malicious code that is triggered when a condition occurs
Time bomb	is a malicious code that is triggered when specified time occurs
Trapdoor	is a feature in a program by which someone can access the program other than by the obvious
Worm	Propagates copies of itself through a network
Rabbit	Replicates itself without limit to exhaust resources

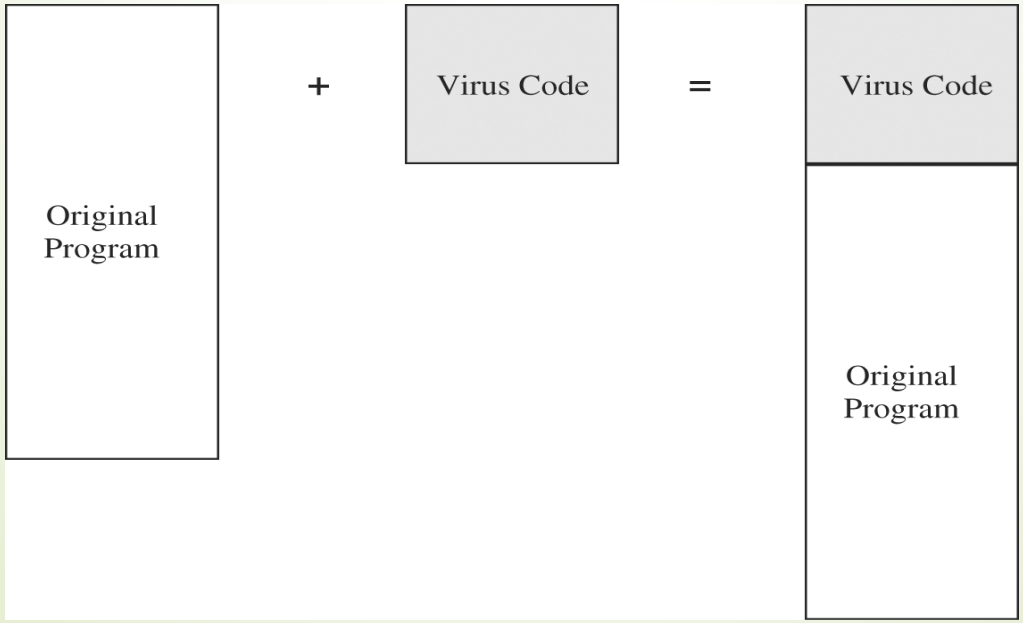
Kinds of Malicious Code

- **Transient:** runs when its attached program executes and terminates when its attached program ends.
- **Resident**: locates itself in memory so that it can remain active even after its attached program ends.

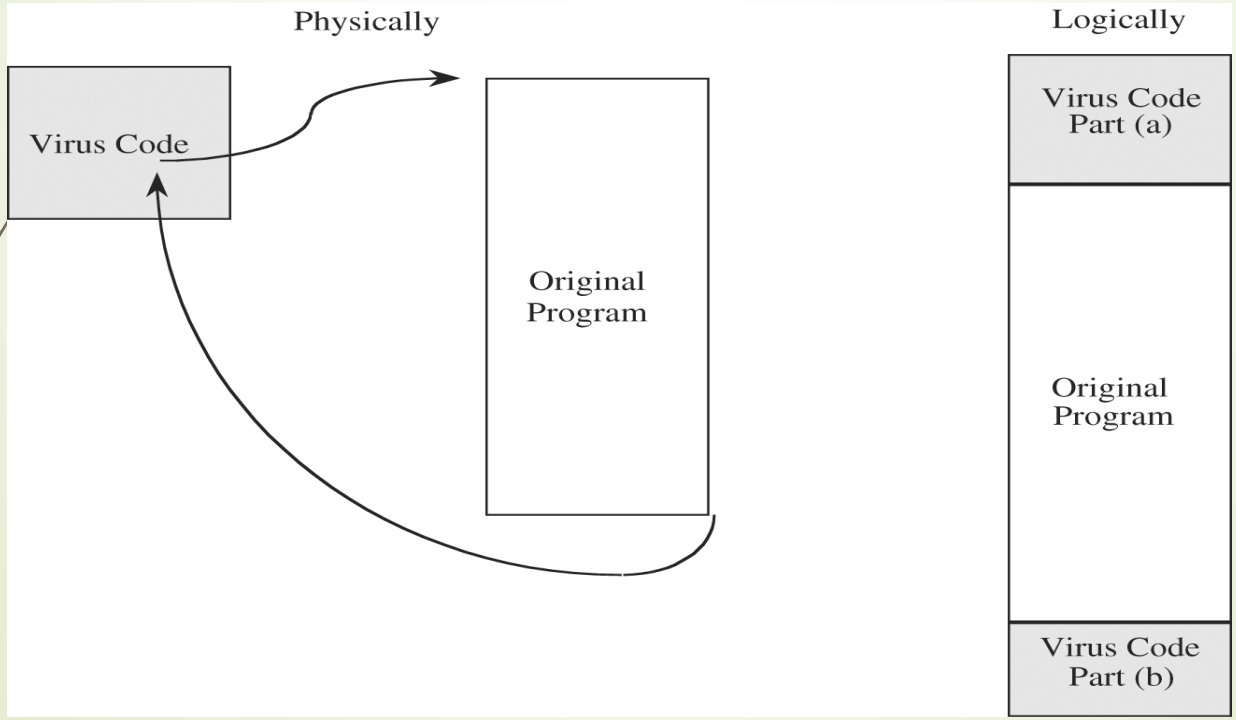
Virus Effect	How It Is Caused
Remain in memory	• Intercept interrupt by modifying interrupt handler address table • Load self in nontransient memory area
Infect disks	• Intercept interrupt • Intercept operating system call (to format disk, for example) • Modify system file • Modify ordinary executable program
Conceal self	• Intercept system calls that would reveal self and falsify result • Classify self as "hidden" file
Spread infection	• Infect boot sector • Infect systems program • Infect ordinary program • Infect data ordinary program reads to control its execution
Prevent deactivation	• Activate before deactivating program and block deactivation • Store copy to reinfect after deactivation

How Viruses Attach themselves into programs

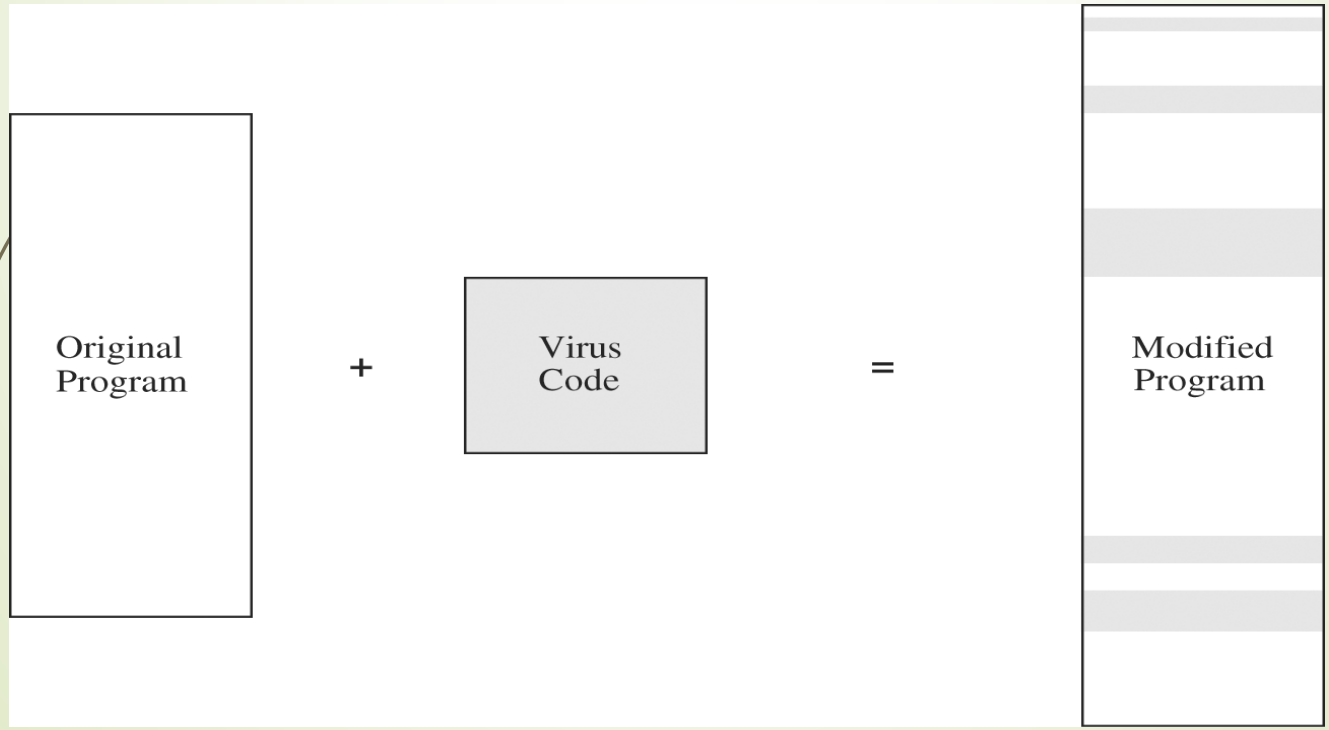
- **Appended Viruses:** A program virus attaches itself to a program; then, whenever the program is run, the virus is activated. This kind of attachment is usually easy to program.



- **<u>Viruses That Surround a Program: A</u>** virus that runs the original program but has control before and after its execution.



- **<u>Integrated Viruses and Replacements</u>:** A virus replaces some of its target, integrating itself into the original code of the target.



Document Viruses

Document virus is implemented within a formatted document, such as a written document, a database, a slide presentation, a picture, or a spreadsheet.

- These documents are highly structured files that contain both data and commands.
- The commands are part of a rich programming language, including macros, variables and procedures, file accesses, and even system calls.

Homes for Viruses

The virus writer may find these qualities appealing in a virus:

- It is hard to detect.
- It is not easily destroyed or deactivated.
- It spreads infection widely.
- It can re-infect its home program or other programs.
- It is easy to create.
- It is machine independent and operating system independent.

***** Few viruses meet all these criteria**

Homes for Viruses (cont.)

- **One-Time Execution:** The majority of viruses today execute only once, spreading their infection and causing their effect in that one execution.
- **Boot Sector Viruses:** Change control on the OS instructions, when a computer is started.
- **Memory-Resident Viruses:** Virus writers also like to attach viruses to resident code because the resident code is activated many times while the machine is running.
- **Other Homes for Viruses:** application programs and libraries.

Virus Signatures

A virus cannot be completely invisible. Code must be stored somewhere and the code must be in memory to execute. Moreover, the virus executes in a particular way, using certain methods to spread. Each of these characteristics yields a telltale pattern, called a **signature.**

Virus scanner looks for the virus signature to detect its existence.

polymorphic virus changes its appearance.

4- Program development controls against malicious code and vulnerabilities

Software engineering principles and practices to Prevent Virus Infection

- ***specify* the system**, by capturing the requirements and building a model of how the system should work from the users' point of view
- ***design* the system**, by proposing a solution to the problem described by the requirements and building a model of the solution *implement* the system, by using the design as a blueprint for building a working solution
- ***test* the syste**m, to ensure that it meets the requirements and implements the solution as called for in the design
- ***review* the system at various stages**, to make sure that the end products are consistent with the specification and design models
- ***document* the system**, so that users can be trained and supported
- ***manage* the system**, to estimate what resources will be needed for development and to track when the system will be done
- ***maintain* the system**, tracking problems found, changes needed, and changes made, and evaluating their effects on overall quality and functionality

Software engineering principles and practices to Prevent Virus Infection

- The only way to prevent the infection of a virus is not to receive executable code from an infected source.

- **Some technique that help:**
  - Use only commercial software acquired from reliable, well-established vendors.
  - Test all new software on an isolated computer.
  - Open attachments only when you know them to be safe.
  - Make a recoverable system image and store it safely.
  - Make and retain backup copies of executable system files.
  - Use virus detectors (often called virus scanners) regularly and update them daily.

5- Controls to protect against program flaws in execution

Controls Against Program Threats
Developmental Controls

- Modularity, Encapsulation, and Information Hiding.
- Peer Reviews.
- Independent Testing.
- Good design.
- Configuration Management.
- Proofs of Program Correctness.

THANK YOU

Reference:

1. Textbook: Security in Computing, 5th Edition , 2015 by Charles P. Pfleeger.
2. Stallings W, Brown L, Bauer MD, Bhattacharjee AK. Computer security: principles and practice. Upper Saddle River, NJ, USA: Pearson Education; 2012.
3. Internet resources.