Unit 4
Program Security

Topics to be covered

- Secure programs
- Non malicious Program
- Errors
- Viruses and other malicious code
- Types of viruses
- Attack mechanism of viruses
- Targeted Malicious Code
- Controls Against Program Threats.

Program Security

- Its our first step on how to apply security to computing
- Protecting programs is the heart of computer security
  - All kinds of programs, from apps via OS, DBMS, networks
- Issues:
  - How to keep programs free from flaws
  - How to protect computing resources from programs with flaws

What is Program Security?

- Depends on whom you ask
  - user
  - programmer
  - manager - conformance to all specifications

- Developmental criteria for program security include:
  - Correctness of security & other requirements
  - Correctness of implementation
  - Correctness of testing

Fault tolerance terminology

- Error - may lead to a fault
- Fault - cause for deviation from intended function
- Failure - system malfunction caused by fault

- Faults - seen by ”insiders” (e.g., programmers)
- Failures - seen by “outsiders” (e.g., independent testers, users)

Fault tolerance terminology

- Error/fault/failure example:
  - Programmer's indexing error, leads to buffer overflow fault
  - Buffer overflow fault causes system crash (a failure)

- Two categories of faults w.r.t. duration
  - Permanent faults
  - Transient faults – can be much more difficult to diagnose

Secure Programs

- What is a secure program?
- Everyone has their own requirement of "being secure"
- Part of assessing software quality
- Does it meet security requirements in specification?
- In general, we often look at quantity and types of faults for evidence of security.

Fixing Faults

- Finding lots of faults in software early.
  - NOT GOOD.
- Early approaches were “Dig” and then “Patch”
  - NOT GOOD.

- Repairing with a patch is a narrow focus area.
- Patches can cause other problems.
  - Non obvious side effects
  - Fix one place – fails another
  - Performance or function suffers

Flaws

- Comparing program requirements with behavior to identify any unexpected behavior is called as **program security flaw**
- Flaw is either a fault or failure
- Vulnerability is a class of flaws (e.g. buffer overflow)
- Need to determine how to prevent harm caused by possible flaws

- Hindrances for eliminating program security flaws
  - How do we test for what a program shouldn't do?
  - Programming and software engineering techniques evolve more rapidly than computer security techniques

Types of Flaws

- **Intentional**
  - Malicious
  - Nonmalicious
- **Inadvertent**
  - Validation error (incomplete / inconsistent): permission checks
  - Domain error: controlled access to data
  - Serialization and aliasing: program flow order
  - Inadequate identification and authentication: basis for authorization
  - Boundary condition violation: failure on first or last case
  - Other exploitable logic errors

Non-malicious Program Errors

- These errors cause program malfunctions but do not lead to more serious security vulnerabilities

- We consider three classic error types:
  1. Buffer overflows
  2. Incomplete mediation
  3. Time-to-check to Time-to-Use Errors

Nonmalicious Program Errors

- **Buffer Overflows**
- A buffer is a space in which data can be held. A buffer's capacity is finite.
- The programmer must declare the buffer's max. size so that the compiler can set aside that amount of space.

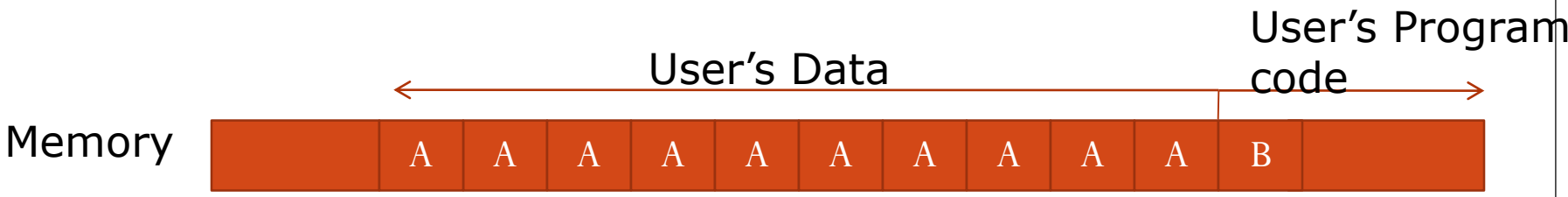
Nonmalicious Program Errors

- Buffer may overflow into (and change):
  - User's own data space
  - User's program area
  - System data space
  - System program area

Nonmalicious Program Errors

- **Buffer Overflows:**

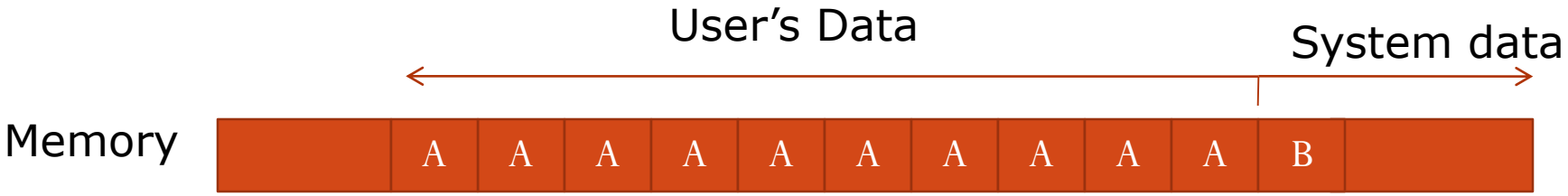
a) Affects user's data



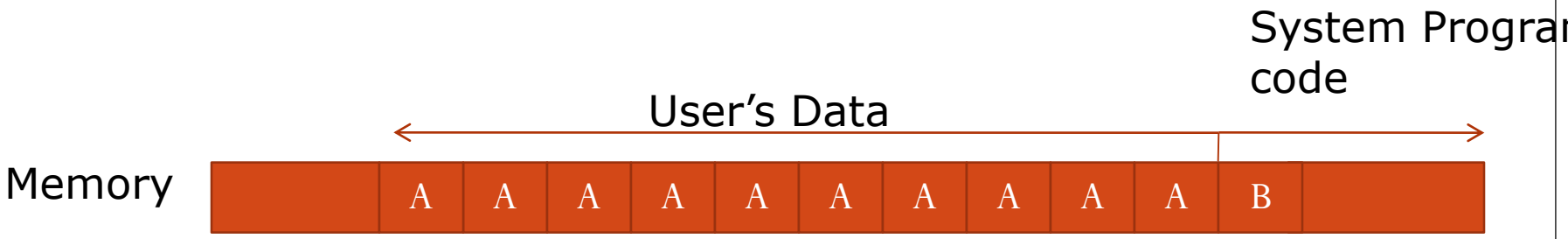
a) Affects user's code

Nonmalicious Program Errors

- **Buffer Overflows:**



a) Affects system data



a) Affects system code

Nonmalicious Program Errors

- Buffer Overflows Security Implication
  - Attacker replaces code in the system space and takes control back from the operating system
  - Attacker uses the stack pointer or return register to execute other codes

Nonmalicious Program Errors

- Incomplete Mediation (data checking)
- Attackers are exploiting it to cause security problems.
  - Supplying the wrong type of data being requested.
  - Supplying the wrong length of data being requested.
  - Problem
    - System Fails
    - Supply of Bad Data
  - Must be checked by programmer
  - Client side v/s Server Side
    - http://www.somesite.com/subpage/data&parm1=(808)555-1212&parm2=2004Jan01
    - What if parm2 is 1800Jan01 or 2004Feb30…
    - the user could send incorrect data to the server
- Security Implication
  - Easy to exploit

Nonmalicious Program Errors

Time-of-Check to Time-of-Use Errors

- The third programming flaw we investigate involves synchronization.
- To improve efficiency, modern processors and operating systems usually change the order in which instructions and procedures are executed.
- In particular, instructions that appear to be adjacent may not actually be executed immediately after each other, either because of intentionally changed order or because of the effects of other processes in concurrent execution.

Nonmalicious Program Errors

Time-of-Check to Time-of-Use Errors

- Definition:

  Access control is a fundamental part of computer security; we want to make sure that only those who should access an object are allowed that access.

- This flaw concerns mediation that is performed with a "bait and switch" in the middle. It is also known as a serialization or synchronization flaw.

Nonmalicious Program Errors

- Bait-and-switch is a form of fraud used in retail sales but also employed in other contexts.
- First, customers are "baited" by merchants' advertising products or services at a low price, but when customers visit the store, they discover that the advertised goods are not available, or the customers are pressured by salespeople to consider similar, but higher-priced items ("switching").
- Security Implication
  - to avoid checking one action and performing another – use digital signatures and certificates

Time of Check, Time of Use

- ToCToU conditions
  - Can occur during file I/O
  - first checking some object and then using it

Nonmalicious Program Errors

Combinations of Nonmalicious Program Flaws

- These three vulnerabilities are bad enough when each is considered on its own.

- But perhaps the worst aspect of all three flaws is that they can be used together as one step in a multistep attack.

Viruses and Other Malicious Code

- When was the last time you saw a bit?
- Do you know in what form a document file is stored?
- Can you find where a document resides on a disk?
- Can you tell if a game program does anything in addition to its expected interaction with you?
- Which files are modified by a word processor when you create a document?

since users usually do not see computer data directly, malicious people can make programs serve as vehicles to access and change data and other programs.

Malicious Code

- Malicious code is the general name for unanticipated or undesired effects in programs or program parts caused by an agent intent on damage.

- The agent is the writer of the program or a person who causes its distribution.

Kinds of Malicious Code

- Virus – code that attaches to another program and copies itself to other programs
  1. Transient virus – life depends on life of its host
  2. Resident virus – locates inside memory
- Trojan Horse – malicious effect is hidden from user (Ex: login script)
- Logic viruses – triggered by an event and goes off when specific condition occur
- Time viruses – triggered by a time or date

Kinds of Malicious Code

- Trapdoor (backdoor) – feature that allows access to program other than through normal channels
- Worm – program that spreads copies of itself through a network, can be stand alone program
- Rabbit – virus/worm that self-replicates without bound

Kinds of Malicious Code

Code type	**Characteristics**
Virus	Attaches itself to program and propagate copies of itself to other programs
Trojan horse	Contains unexpected additional functionality
Logic bomb	Triggers action when condition occurs
Time bomb	Triggers action when specified time occurs
Trapdoor(Back door)	Allows unauthorized access to functionality
Worm	Propagates copies of itself through network
Rabbit	Replicates itself without limit to exhaust resources

Study Assignment

Based on the kinds of MALICIOUS CODE identify two examples of each type and explain in details.

How do Viruses Attach

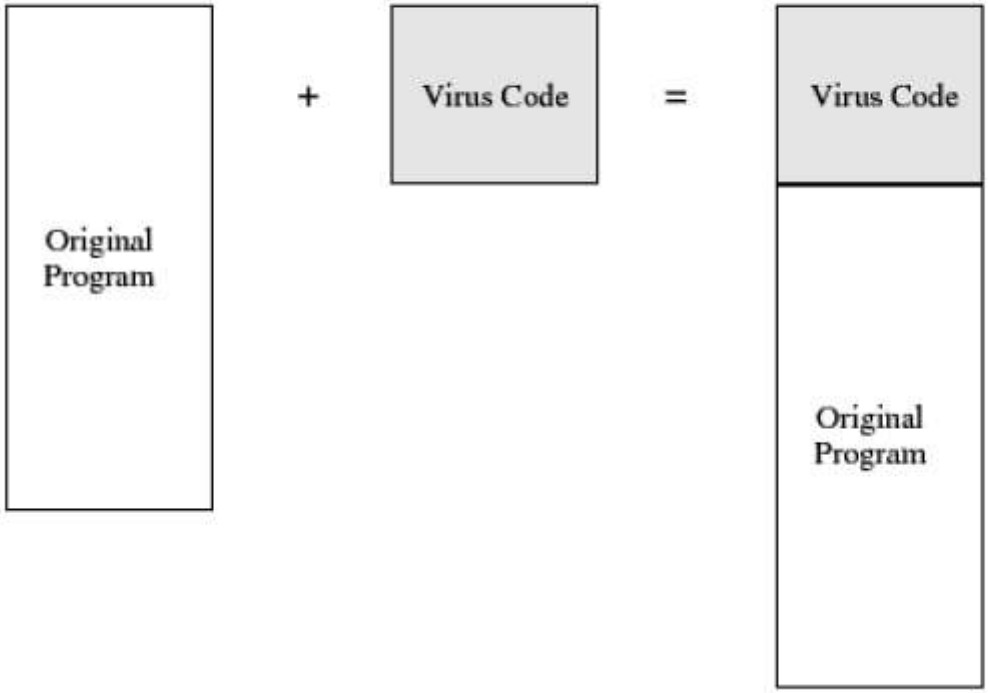
- A virus is attached to a “program”
- The virus is activated by executing the program
- Most common viruses today are attached to e-mail; when the attachment is opened, virus is active

- Three ways:
  1. Appended
  2. Surrounds programs
  3. Integrated viruses and replacements

Appended Viruses

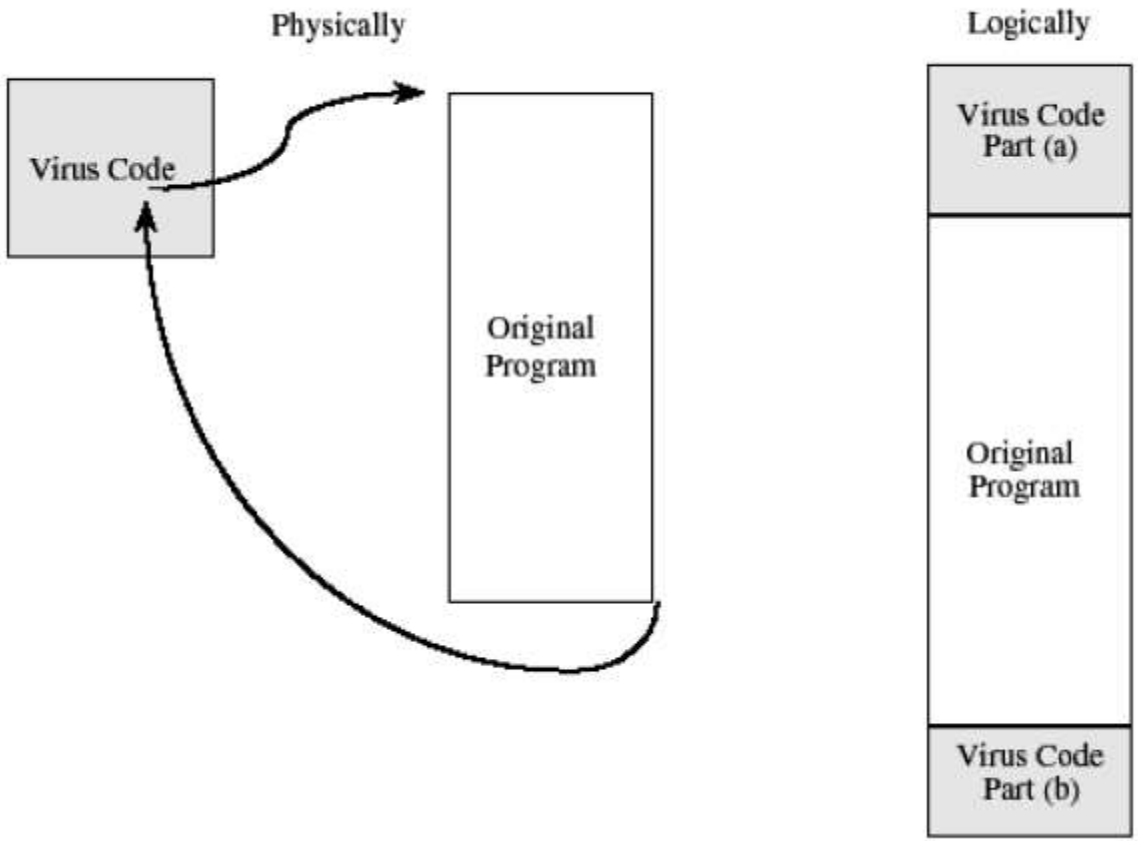
- A program virus attaches itself to a program; then, whenever the program is run, the virus is activated. This kind of attachment is usually easy to program.
- This kind of attachment is simple and usually effective.
- The virus writer does not need to know anything about the program to which the virus will attach, and often the attached program simply serves as a carrier for the virus.

Appended virus example



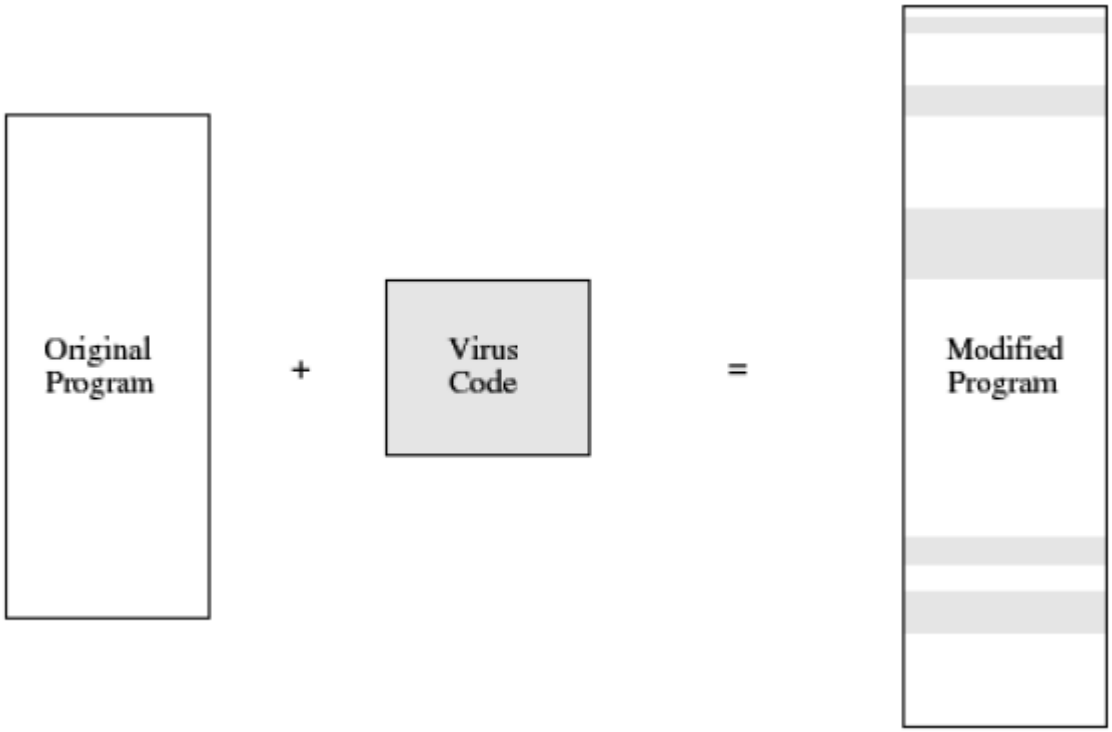
Viruses That Surround a Program

- An alternative to the attachment is a virus that runs the original program but has control before and after its execution.


Physically
Virus Code
Original Program
Logically
Virus Code Part (a)
Original Program
Virus Code Part (b)

Integrated Viruses and Replacements

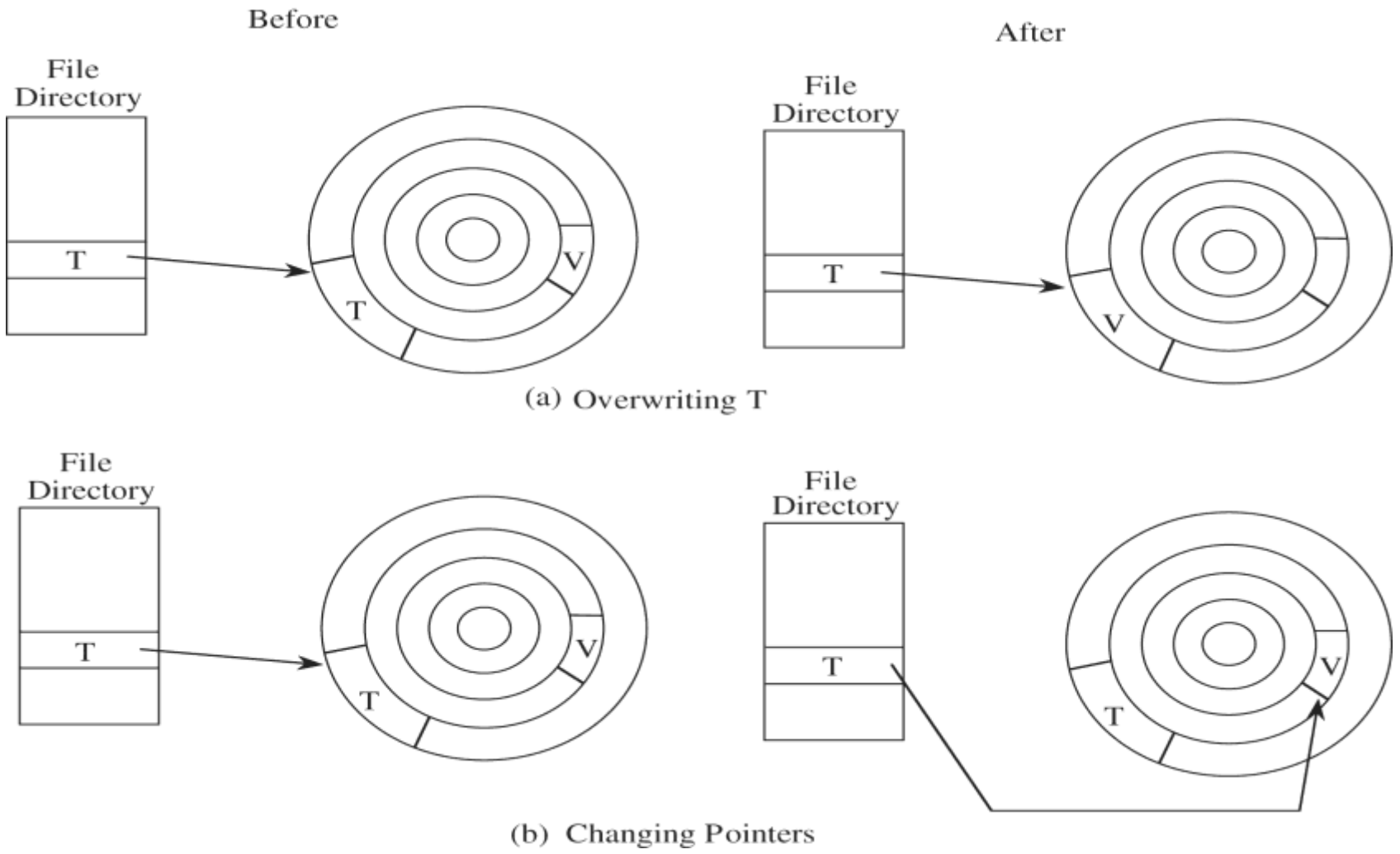
- A third situation occurs when the virus replaces some of its target, integrating itself into the original code of the target.
- The virus writer has to know the exact structure of the original program to know where to insert which pieces of the virus.
- Finally, the virus can replace the entire target, either mimicking the effect of the target or ignoring the expected effect of the target and performing only the virus effect


Original Program
+
Virus Code
=
Modified Program

How viruses gain control

- The virus V has to be invoked instead of the target (T).
- Virus (V) modify/overwrite the program (T) in storage.
- Virus can change file pointer in file table for itself to be located instead of T.

How viruses gain control



Home for viruses

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

Types of viruses

1. One time execution
2. Boot sector viruses
3. Memory resident viruses
4. Document viruses

One-Time Execution

- The majority of viruses today execute only once, spreading their infection and causing their effect in that one execution.
- A virus often arrives as an e-mail attachment of a document virus.
- It is executed just by being opened.

Boot Sector Viruses

- A special case of virus attachment
- When a computer is started, control begins with firmware that determines which hardware components are present, tests them, and transfers control to an operating system.
- The operating system is software stored on disk. Code copies the operating system from disk to memory and transfers control to it; this copying is called the bootstrap (often boot) load.

Boot Sector Viruses

- To allow for change, expansion, and uncertainty, hardware designers reserve a large amount of space for the bootstrap load.
- The boot sector on a PC is slightly less than 512 bytes, but since the loader will be larger than that, the hardware designers support "chaining," in which each block of the bootstrap is chained to (contains the disk location of) the next block.
- The virus writer simply breaks the chain at any point, inserts a pointer to the virus code and reconnects the chain after the virus has been inserted.

Memory-Resident Viruses

- Some parts of the operating system and most user programs execute, terminate, and disappear, with their space in memory being available for anything executed later.
- For very frequently used parts of the operating system and for a few specialized user programs, it would take too long to reload the program each time it is needed.
- Such code remains in memory and is called "**resident**" code.
- Virus writers like to attach viruses to resident code because the resident code is activated many times while the machine is running. Each time the resident code runs, the virus does too.

Document Viruses

- Currently, the most popular virus type is what we call the document virus, which is implemented within a formatted document, such as a written document, a database, a slide presentation, or a spreadsheet.
- These documents are highly structured files that contain both data (words or numbers) and commands (such as formulas, formatting controls, links).

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/characteristics to spread.
- Each of these characteristics yields a pattern, called a **signature,** that can be found by a program that knows to look for it.

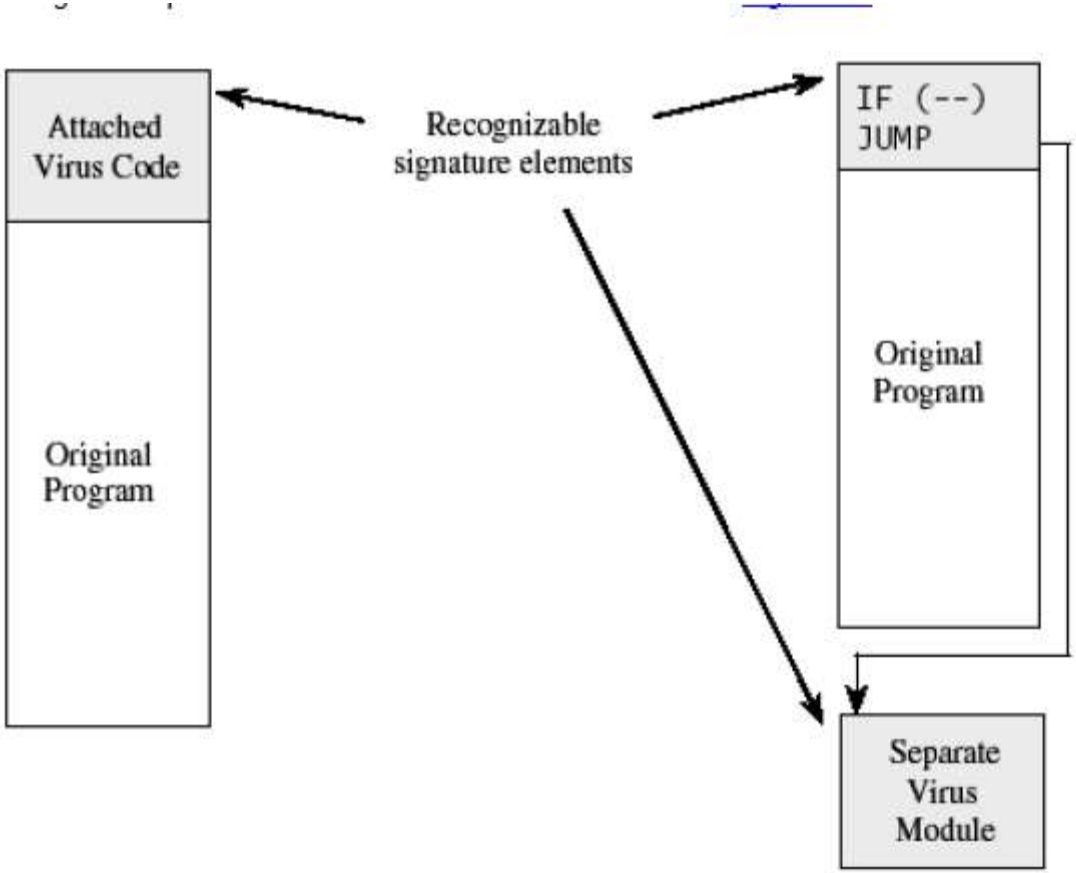
Virus Signatures

- The virus's signature is important for creating a program, called a **virus scanner,** that can automatically detect and, in some cases, remove viruses.
- The scanner searches memory and long-term storage, monitoring execution and watching for the signatures of viruses.
- When the scanner recognizes a known virus's pattern, it can then block the virus, inform the user, and deactivate or remove the virus.
- However, a virus scanner is effective only if it has been kept up-to-date with the latest information on current viruses

Storage Patterns

- Most viruses attach to programs that are stored on media such as disks.
- The attached virus piece is invariant, so that the start of the virus code becomes a detectable signature.
- The attached piece is always located at the same position relative to its attached file.

- A virus may attach itself to a file, in which case the file's size grows.
- Or the virus may obliterate/destroyed all or part of the underlying program, in which case the program's size does not change but the program's functioning will be impaired. The virus writer has to choose one of these detectable effects.
- The virus scanner can use a code or checksum to detect changes to a file. It can also look for suspicious patterns, such as a JUMP instruction as the first instruction of a system program


Attached
Virus Code
Original
Program
Recognizable
signature elements
IF (--)
JUMP
Original
Program
Separate
Virus
Module

Execution Patterns

- A virus writer may want a virus to do several things at the same time, namely, spread infection, avoid detection, and cause harm.
- Unfortunately, many of these behaviours are perfectly normal and might otherwise go undetected. For instance, one goal is modifying the file directory; many normal programs create files, delete files, and write to storage media. Thus, there are no key signals that point to the presence of a virus.

Virus effect	How it is caused
Attach to executable program	Modify file directory Write to executable program file
Attach to data or control file	Modify directory, rewrite data, append to data, append data to self
Remain in memory	Intercept/ interrupt by modifying interrupt handler address table. Load self in nontransient memory area
Infect disks	Intercept interrupt Intercept OS call (eg., to format disk) Modify system file Modify ordinary executable program
Conceal self	Intercept system calls that would reveal self and falsify result
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

Transmission Patterns

- A virus is effective only if it has some means of transmission from one location to another. As we have already seen, viruses can travel during the boot process, by attaching to an executable file or travelling within data files.
- The travel itself occurs during execution of an already infected program. Since a virus can execute any instructions a program can, virus travel is not confined to any single medium or execution pattern.

Polymorphic Viruses

- A virus that can change its appearance is called a polymorphic virus. (*Poly means "many" and morph means "form".) A two-form polymorphic virus can be handled easily as two independent viruses.*
- Therefore, the virus writer intent on preventing detection of the virus will want either a large or an unlimited number of forms so that the number of possible forms is too large for a virus scanner to search for.

Prevention of Virus Infections

- 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 daily and update them regularly

Truths and Misconceptions about viruses

- Viruses can infect only Microsoft Windows systems – False
- Viruses can modify "hidden" or "read-only" files – True
- Viruses can appear only in data files, or only in Word documents, or only in programs – False
- Viruses spread only on disks or only in e-mail – False
- Viruses cannot infect hardware – True

- WINE

EXAMPLES

1. **Brain Virus** - Brain (computer virus)

- **Brain** is the industry standard name for a computer virus that was released in its first form in 19 January 1986, and is considered to be the first computer virus for the IBM Personal Computer (IBM PC) and compatibles. Designed by Pakistan

2. Internet Worm - The **Morris worm** or **Internet worm of November 2, 1988**, was one of the oldest computer worms distributed via the Internet, and the first to gain significant mainstream media attention.

EXAMPLES

- Code RED Worm - **Code Red** was a computer worm observed on the Internet on July 15, 2001. It attacked computers running Microsoft's IIS web server. Type – Server jamming worm.
- SQL-Server Slammer - **SQL Slammer** is a 2003 computer worm that caused a denial of service on some Internet hosts and dramatically slowed general Internet traffic.
- Web Bugs (spyware) - Web bugs are tiny (usually a single pixel) transparent image files on web pages that are used to monitor user's online habits.

TARGETED MALICIOUS CODE

- Another class of malicious code is written for a particular system, for a particular application, and for a particular purpose.
- Examples:
- Trapdoor – undocumented entry point to a module
- Salami Attack (Ex. Interest computation). An attack on a computer network which involves the intruder siphoning off small amounts of money from a file and placifile that holds their bank account detailsng them in another file that he or she can access; for example, a.
- Covert Channels: programs that leak information (Ex. Hide data in output). Type of computer security attack that creates a capability to transfer information objects between processes that are not supposed to be allowed to communicate

Trapdoors

A trapdoor is an undocumented entry point to a module. The trapdoor is inserted during code development, perhaps to test the module, to provide "hooks" by which to connect future modifications or enhancements or to allow access if the module should fail in the future.

In addition to these legitimate uses, trapdoors can allow a programmer access to a program once it is placed in production.

Trapdoors

Causes of trapdoors:

- Forget to remove them
- Intentionally leave them in the program for testing
- Intentionally leave them in the program for maintenance of the finished program
- Intentionally leave them in the program as a covert means of access to the component after it becomes an accepted part of a production system.

Salami Attack

- A salami attack merges bits of seemingly insignificant data to yield powerful results.
- For example, programs often disregard small amounts of money in their computations, as when there are fractional pennies as interest or tax is calculated.
- These small amounts are shaved from each computation and accumulated elsewhere, it is unlikely to be noticed for an individual case.
- Salami attack uses siphoning technique. siphoning is a technique used to **"steal" traffic that would normally be directed to another website in search engine results**

Covert Channels

Storage channels:

- They pass information using the presence/absence of objects in storage.
- Example: file lock channel
  - In multiuser systems, files can be locked to prevent two people from writing to the same file at the same time.
  - A covert channel can signal one bit of information by whether or not a file is locked.

Timing channels:

- They pass information by using the speed at which things happen.
- These are shared resource channels in which the shared resource is time.

Controls against program threats

- There are many ways a program can fail and many ways to turn the underlying faults into security failures.
- It is better to focus on prevention than cure; how do we use controls during s/w development to find and eliminate the faults.
- 3 types of controls:
  - Developmental controls
  - Operating system
  - administrative

Controls against program threats

- **Developmental controls:**
- Many controls can be applied during s/w development to fix problems.
- The nature of s/w development:
  - specify
  - Design
  - Implement
  - Test
  - Review
  - Document the system
  - Manage
  - maintain

Controls against program threats

- **Modularity, Encapsulation and information hiding:**
- A key principle of SE is to create a design or code in small, self-contained units called components/modules, when a system is written this way, it is called **modular**
- If a component is isolated from the effects of other components, then
  - it is easier to trace a problem to the fault that caused it
  - easier to maintain the system
  - Easier to see where vulnerabilities may lie
- This isolation is called **encapsulation**
- **Information hiding** is another characteristic of modular s/w.
- When information is hidden, each component hides its implementation from others, so that when a change is needed, the overall design can remain intact.

Controls against program threats

- **Modularity:**
- Modularization is a process of dividing a task into subtasks.
- The goal is to have each component meet 4 conditions:
- **Single-purpose**: performs one function
- **Small:** consists of an amount of information for which a human can readily grasp both structure and content.
- **Simple**: is of a low degree of complexity so that a human can readily understand the purpose and structure of the module.
- **Independent:** performs a task isolated from other modules.

Controls against program threats

- **Modularity:**
- Advantages:
  - Maintenance
  - Understandability
  - Reuse
  - Correctness
  - Testing

- A modular component has high cohesion and low coupling

Controls against program threats

- **Modularity:**

- **Cohesion**: all elements of a component have a logical and functional reason; every aspect of the component is tied to the component's single purpose.
- A high cohesive component has high degree of focus on purpose
- Low degree cohesion means that component's are unrelated.

- **Coupling**: refers to the degree with which a component depends on other components in the system
- Thus, low coupling is better than tight coupling.

Controls against program threats

- **Encapsulation**:
- It hides a component's implementation details, but it does not necessarily mean complete isolation.
- The sharing is documented so that a component is affected only in known ways by others.
- Sharing is minimized so that fewest interfaces possible are used.
- **Information hiding**:
- Component is a black box with certain well-defined inputs and outputs and well-defined function.
- Other components do not need to know how the module completes its function.

Controls against program threats

- Peer Reviews
- Hazard Analysis – set of systematic techniques to expose potentially hazardous system states
- Testing – unit testing, integration testing, function testing, performance testing, acceptance testing, installation testing, regression testing
- Good Design
  - Using a philosophy of *fault tolerance*
  - Have a consistent *policy* for handling failures
  - Capture the *design rationale* and history
  - Use design patterns

Controls against program threats

- Prediction – predict the risks involved in building and using the system
- Static Analysis – Use tools and techniques to examine characteristics of design and code to see if the characteristics warn of possible faults
- Configuration Management – control changes during development and maintenance
- Analysis of Mistakes
- Proofs of Program Correctness – Can we prove that there are no security holes?

Operating System Controls on Usage of Programs

- Trusted Software – code has been rigorously developed and analyzed
  - Functional correctness
  - Enforcement of integrity
  - Limited privilege
  - Appropriate confidence level
- Mutual Suspicion – Suspicion or a suspicion is a belief or feeling that someone has committed a crime or done something wrong. Assume other program is not trustworthy
- Confinement – limit resources that program can access
- Access Log – list who access computer objects, when, and for how long

Administrative Controls

- Standards of Program Development
  - Standards of design
  - Standards of documentation, language, and coding style
  - Standards of programming
  - Standards of testing
  - Standards of configuration management
  - Security Audits
- Separation of Duties according to administrative and operational level managers and workers.