Malicious Code - Malware

LECTURE 8

Malware – Viruses, Trojan Horses, and Worms

- **Malicious code** or **malware** (short for MALicious softWARE) is the general name for programs or program parts planned by an agent with malicious intent to cause unanticipated or undesired effects.
- The agent is the program's writer or distributer. Malicious intent distinguishes this type of code from unintentional errors, even though both kinds can certainly have similar and serious negative effects.
- To sum up:
  - **Malicious code can be directed at a specific user or class of users, or it can be anyone.**

Malware – Viruses, Trojan Horses, and Worms

- A **virus** is a program that can replicate itself and pass on malicious code to other nonmalicious programs by modifying them.
- A virus can be either **transient** or **resident**.
  - A **transient virus** has a life span that depends on the life of its host; the virus runs when the program to which it is attached executes, and it terminates when the attached program ends. (During its execution, the transient virus may spread its infection to other programs.)
  - A **resident virus** locates itself in a memory; it can then remain active or be activated as a stand-alone program, even after its attached program ends.
- To sum up:
  - **Virus**: code with malicious purpose; intended to spread.

Malware – Viruses, Trojan Horses, and Worms

- A **worm** is a program that spreads copies of itself through a network.

- The primary difference between a worm and a virus is that a worm operates through networks, and a virus can spread through any medium (but usually uses a copied program or data files). Additionally, the worm spreads copies of itself as a stand-alone program, whereas the virus spreads copies of itself as a program that attaches to or embeds in other programs.
- To sum up:
  - **Worm**: program that spreads copies of itself through a network.

Malware – Viruses, Trojan Horses, and Worms

- A Trojan horse is a malicious code that, in addition to its primary effect, has a second, nonobvious, malicious effect.
- Trojan horse malware slips inside a program undetected and produces unwelcome effects later on.
- As an example of a computer Trojan horse, consider a login script that solicits a user's identification and password, passes the identification information on to the rest of the system for login processing, but also retains a copy of the information for later, malicious use. In this example, the user sees only the login occurring as expected, so there is no reason to suspect that any other, unwelcome action took place.
- To sum up:
  - **Trojan horse**: program with benign apparent effect but second, hidden, malicious effect.

Malware – Viruses, Trojan Horses, and Worms

To remember the differences among these three types of malware, understand that:

- a **Trojan horse** is on a the surface a useful program with extra, undocumented (malicious) features. It does not necessarily try to propagate.
- By contrast, a **virus** is a malicious program that attempts to spread to other computers, as well as perhaps performing unpleasant action on its current host. The virus does not necessarily spread by using a network's properties; it can be spread instead by travelling on a document transferred by a portable device (e.g., the memory stick) or triggered to spread to other, similar file types when a file is opened.
- However, a **worm** requires a network for its attempts to spread itself elsewhere.

Types of Malicious Code

Code Type	Characteristics
Virus	Code that causes malicious behavior and propagates copies of itself to other programs
Trojan horse	Code that contains unexpected, undocumented, additional functionality
Worm	Code that propagates copies of itself through a network; impact is usually degraded performance
Rabbit	Code that replicates itself without limit to exhaust resources
Logic bomb	Code that triggers action when a predetermined condition occurs
Time bomb	Code that triggers action when a predetermined time occurs
Dropper	Transfer agent code only to drop other malicious code, such as virus or Trojan horse
Hostile mobile code agent	Code communicated semi-autonomously by programs transmitted through the web

Types of Malicious Code

Code Type	Characteristics
Script attack, JavaScript, Active code attack	Malicious code communicated in JavaScript, ActiveX, or another scripting language, downloaded as part of displaying a web page
RAT (remote access Trojan)	Trojan horse that, once planted, give access from remote location
Spyware	Program that intercepts and covertly communicates data on the user or the user's activity
Bot	Semi-autonomous agent, under control of a (usually remote) program
Zombie	Code or entire computer under a (usually remote) program
Browser hijacker	Code that changes browser settings, disallows access to certain sites, or redirects browser to others
Trapdoor or backdoor	Code feature that allows unauthorized access to a machine or program; bypasses normal access control and authentication
Scareware	Not code; false warning of malicious code attack

Technical Details – Malicious Code

Four aspects of malicious code infections:

- ***harm*** – how they affect users and systems
- ***transmission and propagation*** – how they are transmitted and replicate, and how they cause further transmission
- ***activation*** – how they gain control and install themselves so that they can reactivate
- ***stealth*** – how they hide to avoid detection

Harm from Malicious Code

We can divide the payload from malicious code into three categories:

- ***Nondestructive***
- ***Destructive***
- ***Commercial or criminal intent***

Without our knowing the mind of the attacker, motive can be hard to determine (however, the third category has an obvious commercial motive).

Harm to Users

Most malicious code harm occurs to the infected computer's data. Here are some real-world examples of malice.

- Hiding the cursor.
- Displaying text or an image on the screen.
- Opening a browser window to web sites related to current activity.
- Sending email to some or all entries in the user's contacts or alias list.
- Opening text documents and changing some instances of "is" to "is not" and vice versa.
- Deleting all files.

Harm to the User's System

Here are some maneuvers by which malware writers conceal their infection.

- Hide the file in a lower-level directory, often a subdirectory created or used by another legitimate program.
- Attach to a critical system file, especially one that is invoked during system startup (to ensure the malware is reactivated).
- Replace (retaining the name of) a noncritical system file.
- Hide copies of the executable code in more than one location.
- Hide copies of the executable code in different locations on different systems so no single eradication procedure can work.
- Modify the system registry so that the malware is always executed or malware detection is enabled.

Damage Estimates

How do you determine the cost or damage of any computer security incident?

The first step is to enumerate the losses.

Knowing the losses and their approximate cost, you can compute the total cost of an incident.

To sum up:

Estimating the cost of an incident is hard. That does not mean the cost is zero or insignificant, just hard to determine.

Transmission and Propagation

A printed copy of code does nothing and threatens no one. Even executable code sitting on a disk does nothing.

- What triggers code to start?
  - For malware to do its malicious work and spread itself, it must be executed to be activated. Fortunately for malware writers but unfortunately for the rest of us, there are many ways to ensure that programs will be executed on a running computer.

Setup and Installer Program Transmission

Example: the SETUP program that you run to load and install a new program on your computer. It may call dozens or hundreds of other programs, some on the distribution medium, some already residing on the computer, some in memory. If any of these programs contains a virus, the virus code could be activated.

How?

Attached File

A more common means of virus activation is in a file attached to an email message or embedded in a file. In this attack, the virus writer tries to convince the victim (the recipient of the message or file) to open the object. Once the viral object is opened (and thereby executed), the activated virus can do its work.

Solution: forcing users to open files on their own rather than having an application do it automatically is a best practice; programs should not perform potentially security-relevant actions without a user's consent.

However, ease-of-use often trumps security, so programs such as browsers, email handlers, and viewers often "helpfully" open files without first asking the user.

Document Viruses

Document virus is implemented within a formatted document, such as written document, a database, a slide presentation, a picture, or a spreadsheet. These documents are highly structured files that contain both data (words or numbers) and commands (such as formulas, formatting control, links). The commands are part of a rich programming language, including macros, variables and procedures, file accesses and even system calls. The writer of a document virus uses any of the features of the programming language to perform malicious actions.

How?

Autorun

Autorun is a feature of operating systems that causes the automatic execution of code based on name or placement.

In Windows, the registry contains several lists of programs automatically invoked at startup, some readily apparent (in the start menu/programs/startup list) and others more hidden (for example, in the registry key software\windows\current_version\run).

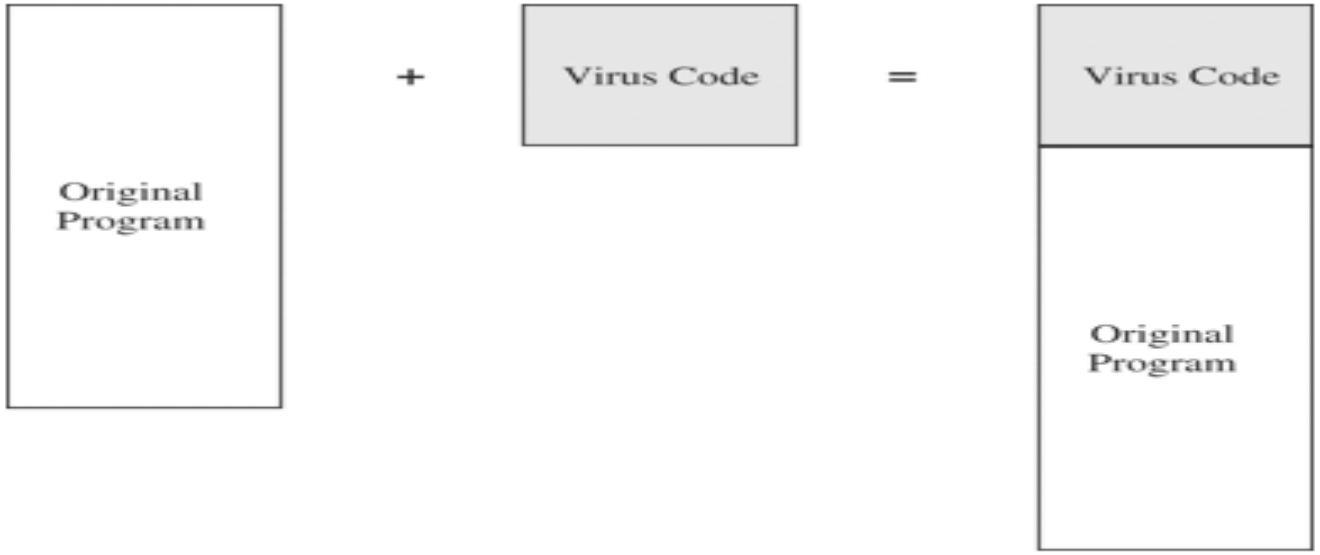
One popular technique for transmitting malware is distribution via flash memory, such as solid state USB memory stick.

Propagation

Since a virus can be rather small, its code can be “hidden” inside other larger and more complicated programs. Two hundred lines of a virus could be separated into one hundred packets of two lines of code and a jump each; these one hundred packets could be easily hidden inside a compiler, a database manager, a file manager, or some other large utility.

Appended Viruses

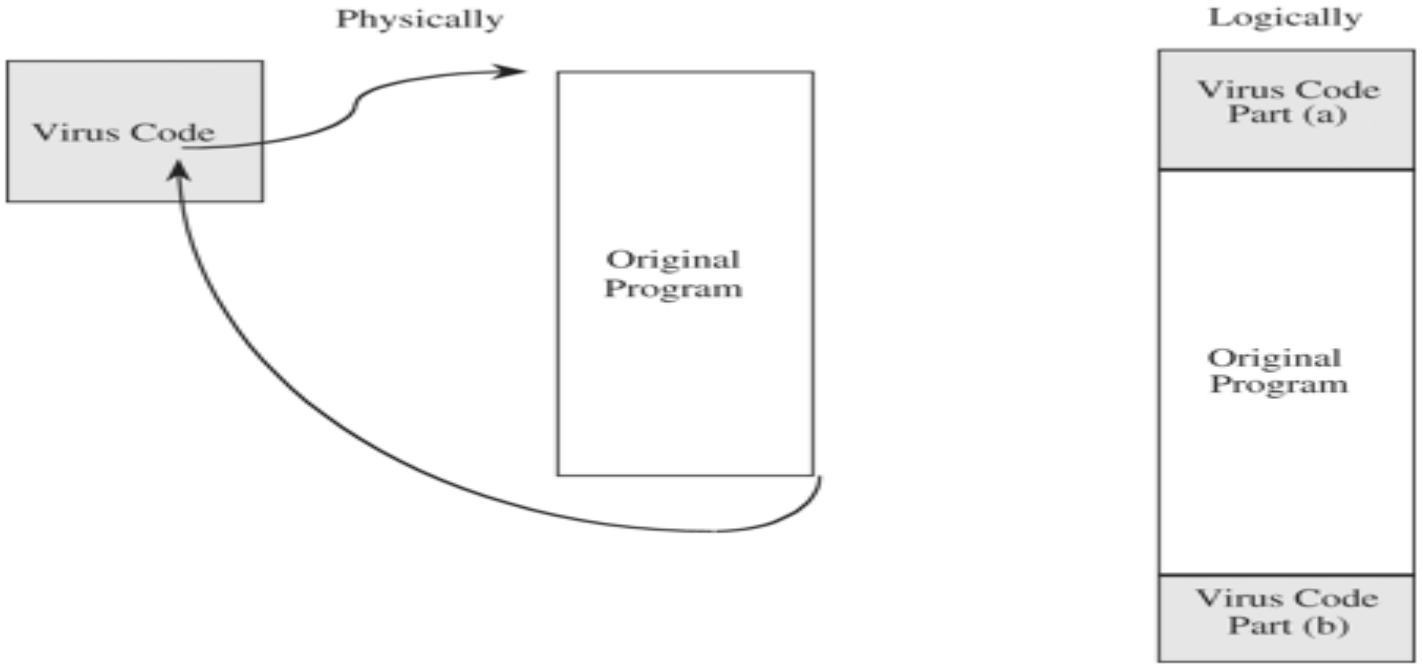
A program virus attaches itself to a program; then, whenever the program is run, the virus is activated. This kind of attachment is usually easy to design and implement.



Virus Appended to a Program

Virus that Surround a Program

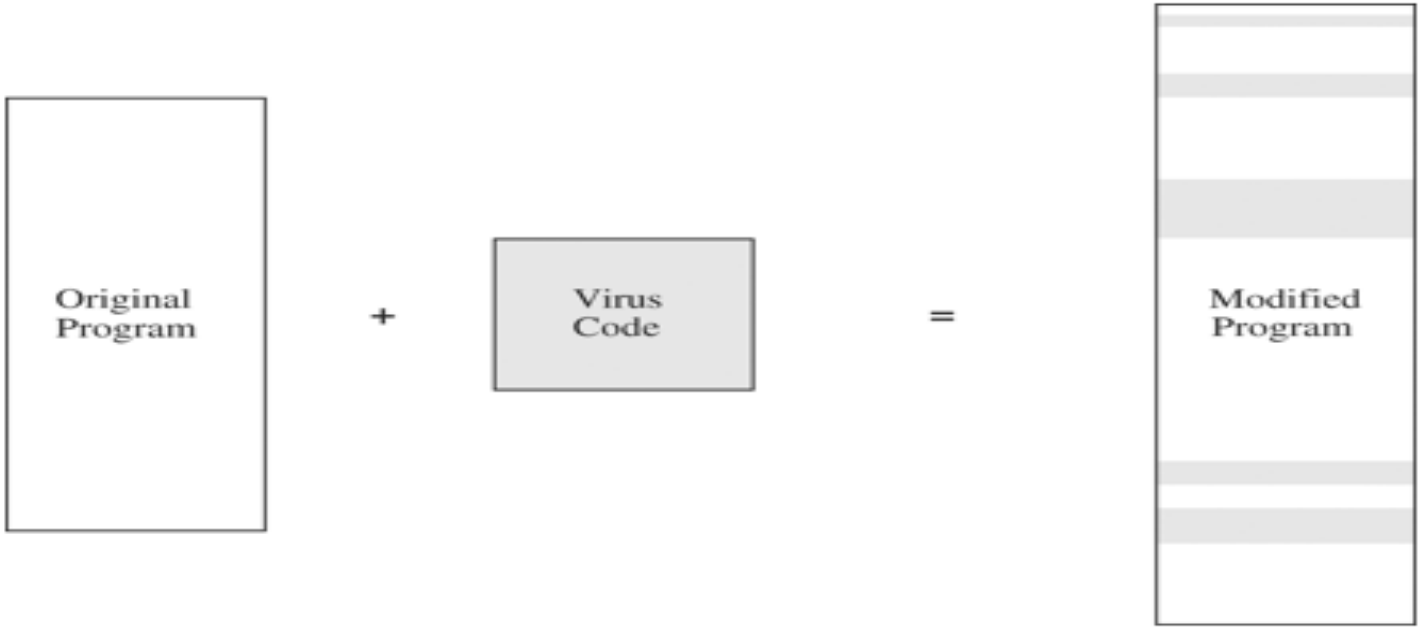
An alternative to the attachment is a virus that runs the original program but has control before and after its execution.



Virus Surrounding a Program

Integrated Viruses and Replacements

In this situation, the virus replaces some of its target, integrating itself into the original code of the target.

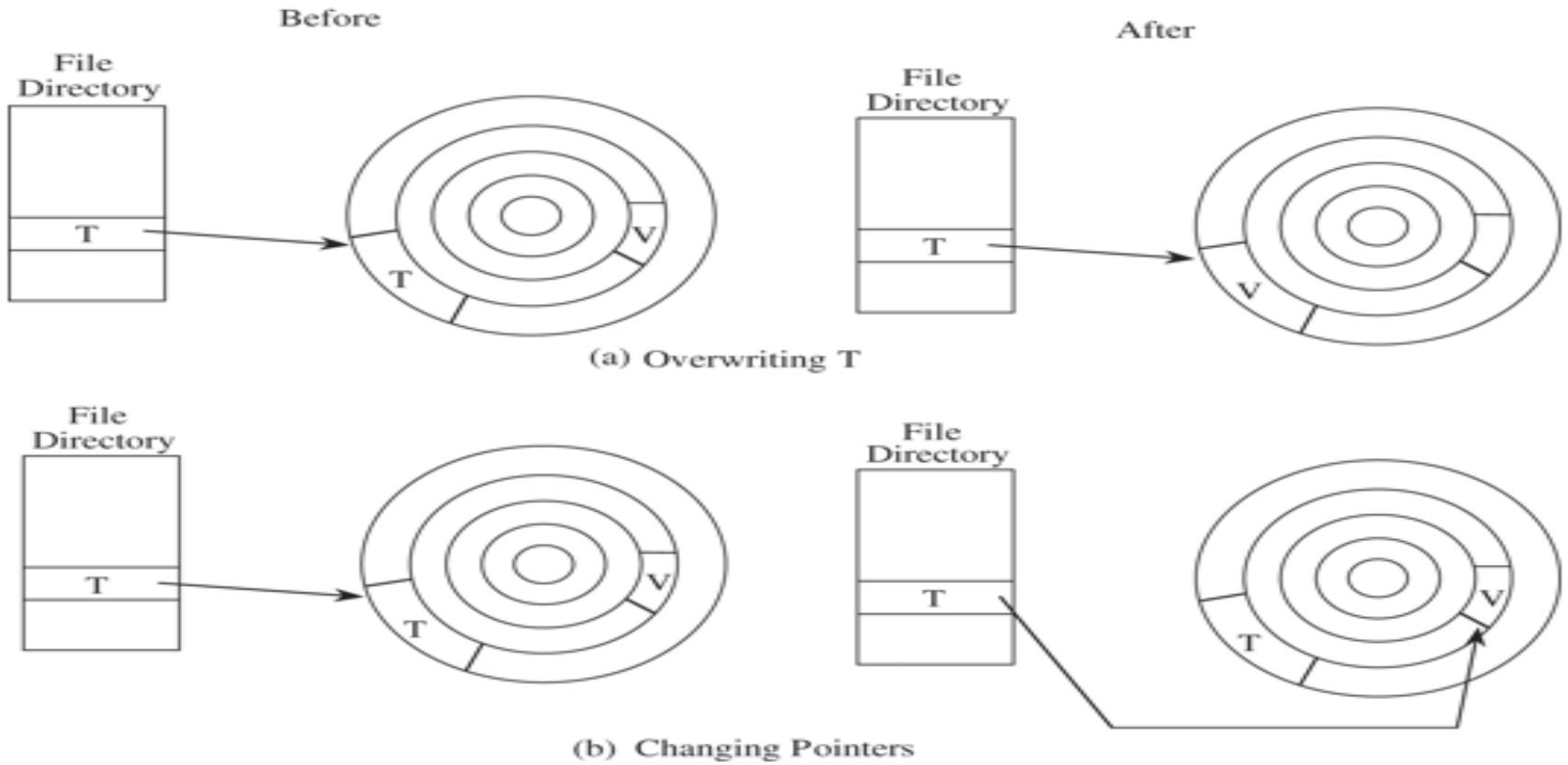


Virus Integrated into a Program

How Malicious Code Gains Control

To gain control of processing, malicious code such as a virus (V) has to be invoked instead of the target (T). Essentially, the virus either has to seem to be T, saying effectively “I am T”, or the virus has to push T out of the way and become a substitute for T, saying effectively “Call me instead of T.”

How Malicious Code Gains Control



Virus Completely Replacing a Program

Embedding: Homes of Malware

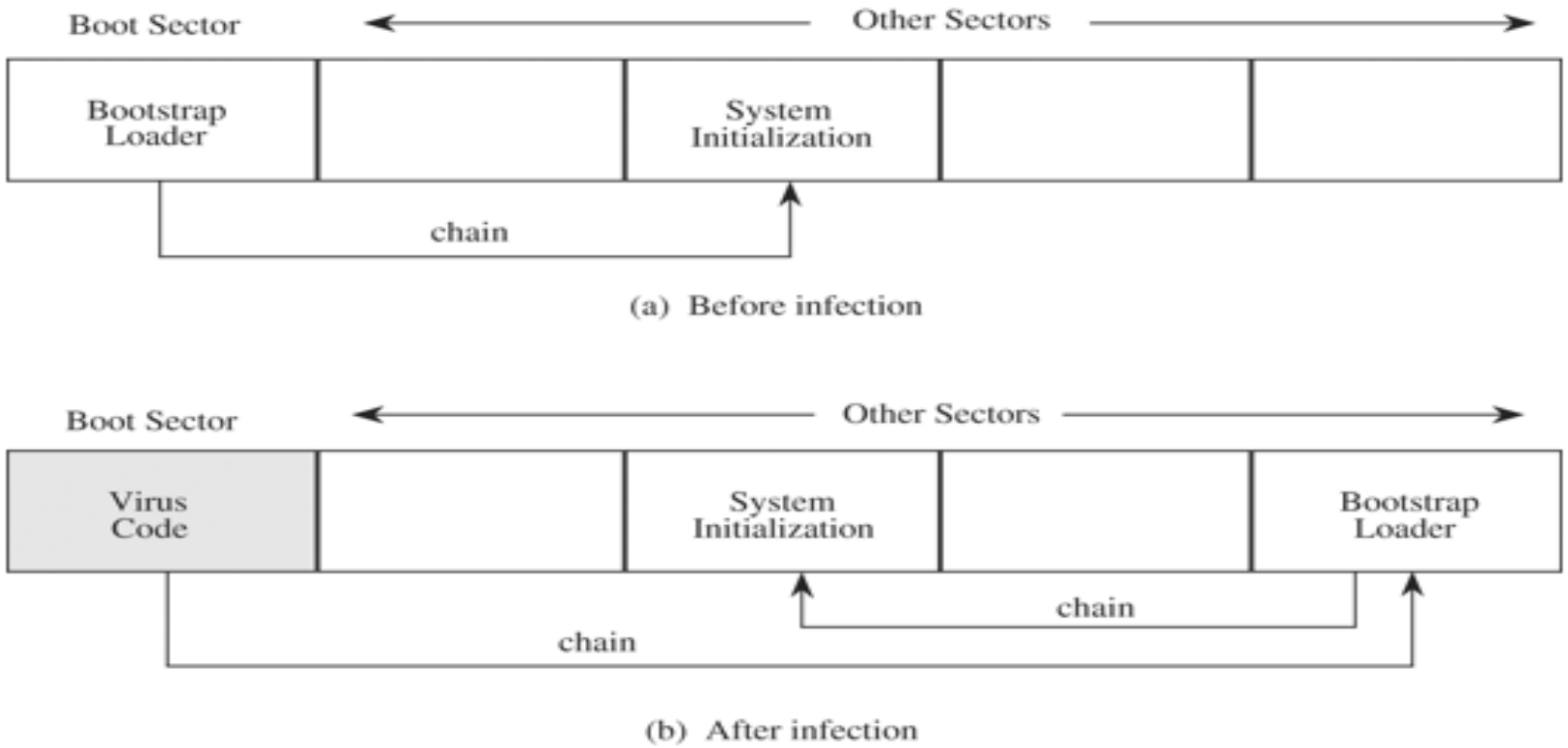
The malware writer may find it appealing to build these qualities into the malware:

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

Boot Sector Viruses

Attackers are interested in creating continuing or repeated harm, instead of just a one-time assault. For continuity the infection needs to stay around and become an integral part of the OS. The easy way to become permanent is to force the harmful code to be reloaded each time the system is restarted.

Boot Sector Viruses



Boot Sector Virus Relocating Code

Stealth

The final objective for a malicious code writer is stealth: avoiding detection during installation, while executing, or even at rest in storage.

- Detection
- Installation Stealth
- Execution Stealth

Stealth in Storage

Antivirus and other malicious code scanners look for patterns because malware writers have the same considerations you would have in developing mass-market software: They want to write one body of code and distribute it to all other victims. That identical code become a pattern on disk for which a scanner can search quickly and efficiently.

Stealth in Storage

Knowing that scanners look for identical patterns, malicious code writers try to vary the appearance of their code in several ways:

- Rearrange the order of modules.
- Rearrange the order of instructions (when order does not affect executions; for example A:=1; B:=2 can be rearranged with no detrimental effect).
- Insert instructions, (such as A:=A), that have no impact.
- Insert random strings (perhaps as constants that are never used).
- Replace instructions with others of equivalent effect, such as replacing A:=B-1 with A:=B+(-1) or A:=B+2-1.
- Insert instructions that are never executed (for example, in the *else* part of a conditional expression that is always true).

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.

Virus Effect	How It Is Caused
Attach the 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 itself
Remain in memory	• Intercept interrupt by modifying interrupt handler address table • Load itself in nontransient memory area

Execution Patterns

Virus Effect	How It Is Caused
Infect disks	• Intercept interrupt • Intercept OS 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 system 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

Countermeasures for Users

There are several techniques for building a reasonable safe community for electronic contact, including the following:

- Use only commercial software acquired from reliable, well-established vendors.
- Test all new software on an isolated computer.
- Open attachments – and other potentially infected data files – only when you know them to be safe.
- Install software - and other potentially infected data files – only when you really know them to be safe.
- Recognize that any web site can be potentially harmful.
- Make a recoverable system image and store it safely.
- Make and retain backup copies of executable system files.

Virus Detectors

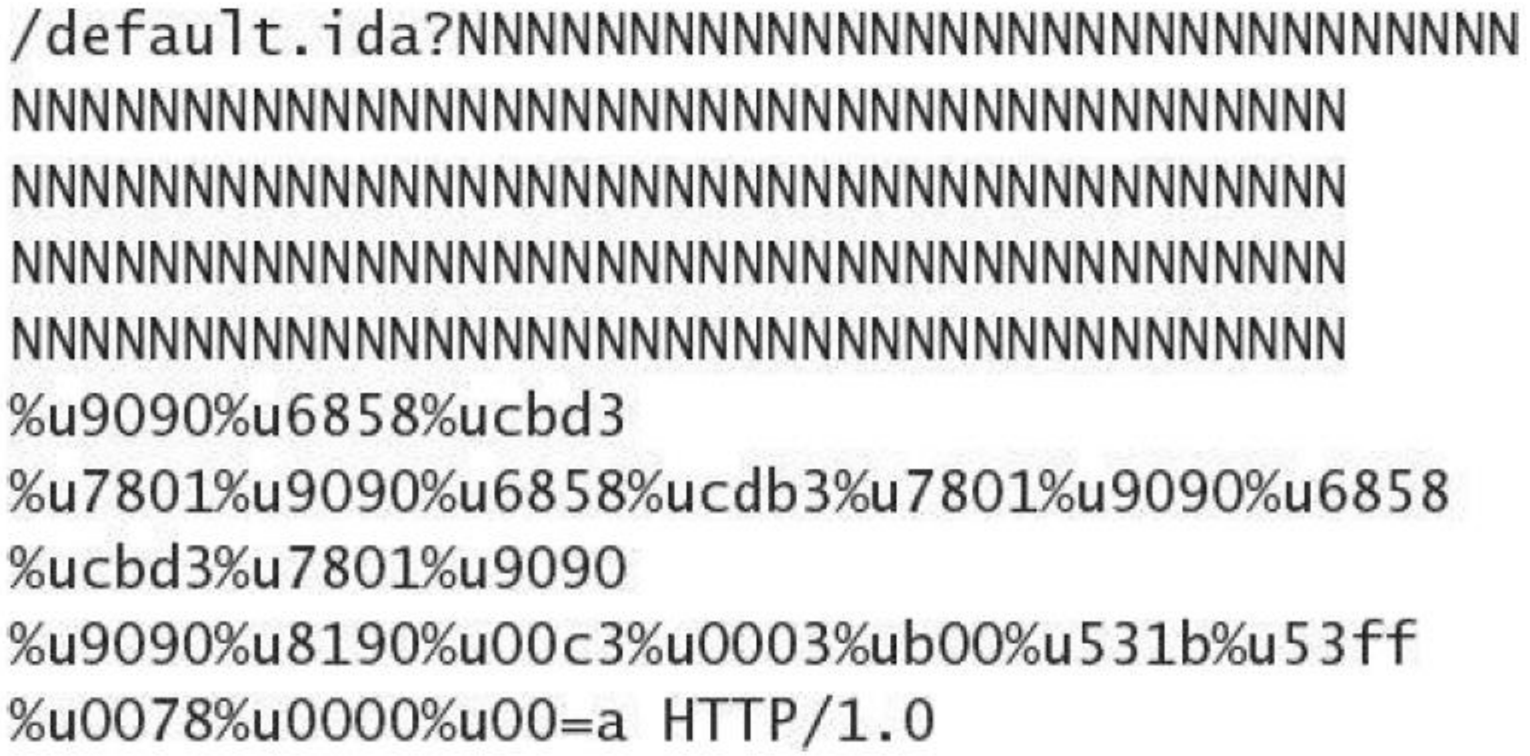
Virus scanners are tools that look for signs of malicious code infection. Most such tools look for a signature or fingerprint, a telltale pattern in programs files or memory. Detection tools are generally effective, meaning that they detect most examples of malicious code that are at most somewhat sophisticated.

Detection tools do have two major limitations:

- detections tools are necessarily retrospective, looking for patterns of known infections.
- patterns are necessarily static.

Virus Signatures

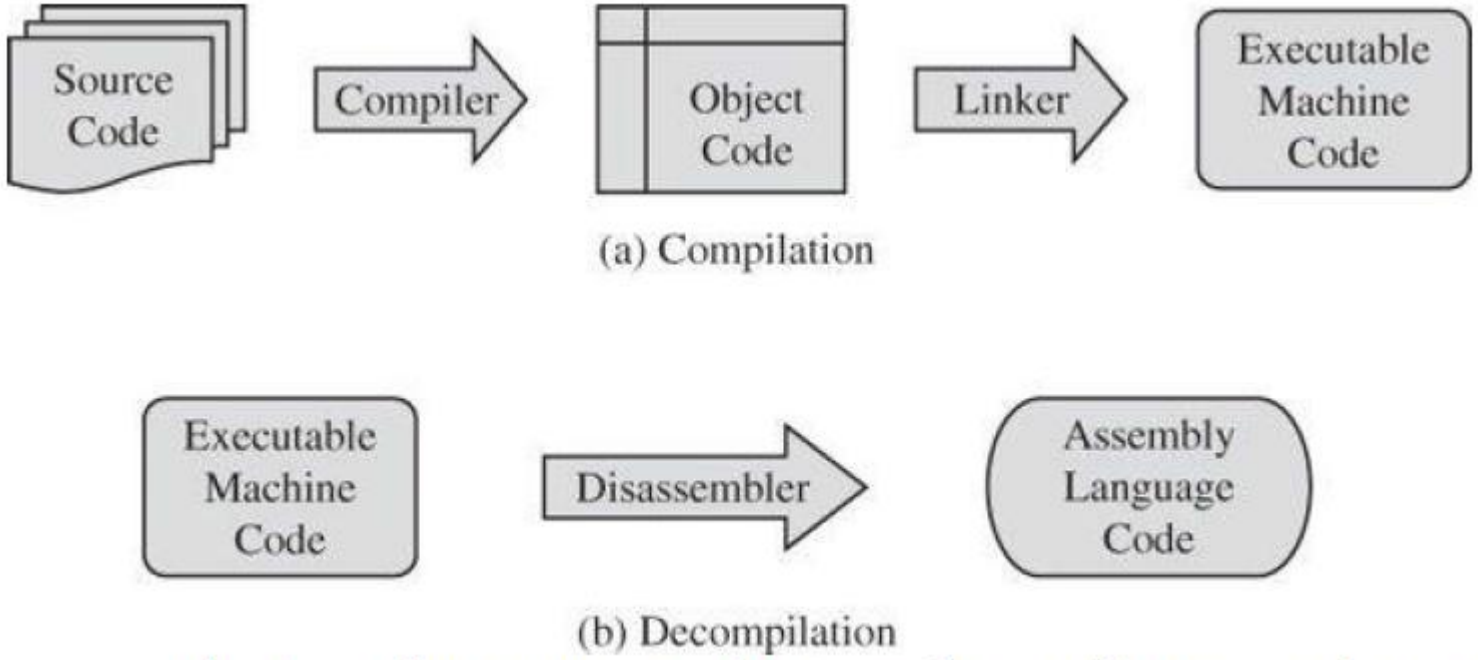
Example, a scanner looking for signs of the Code Red worm can look for a pattern containing the following characters:



```
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
%u9090%u6858%ucbd3
%u7801%u9090%u6858%ucdb3%u7801%u9090%u6858
%ucbd3%u7801%u9090
%u9090%u8190%u00c3%u0003%ub00%u531b%u53ff
%u0078%u0000%u00=a HTTP/1.0
```

Code Analysis

Another approach to detecting an infection is to analyze the code to determine what it does, how it propagates and perhaps even where it originated.

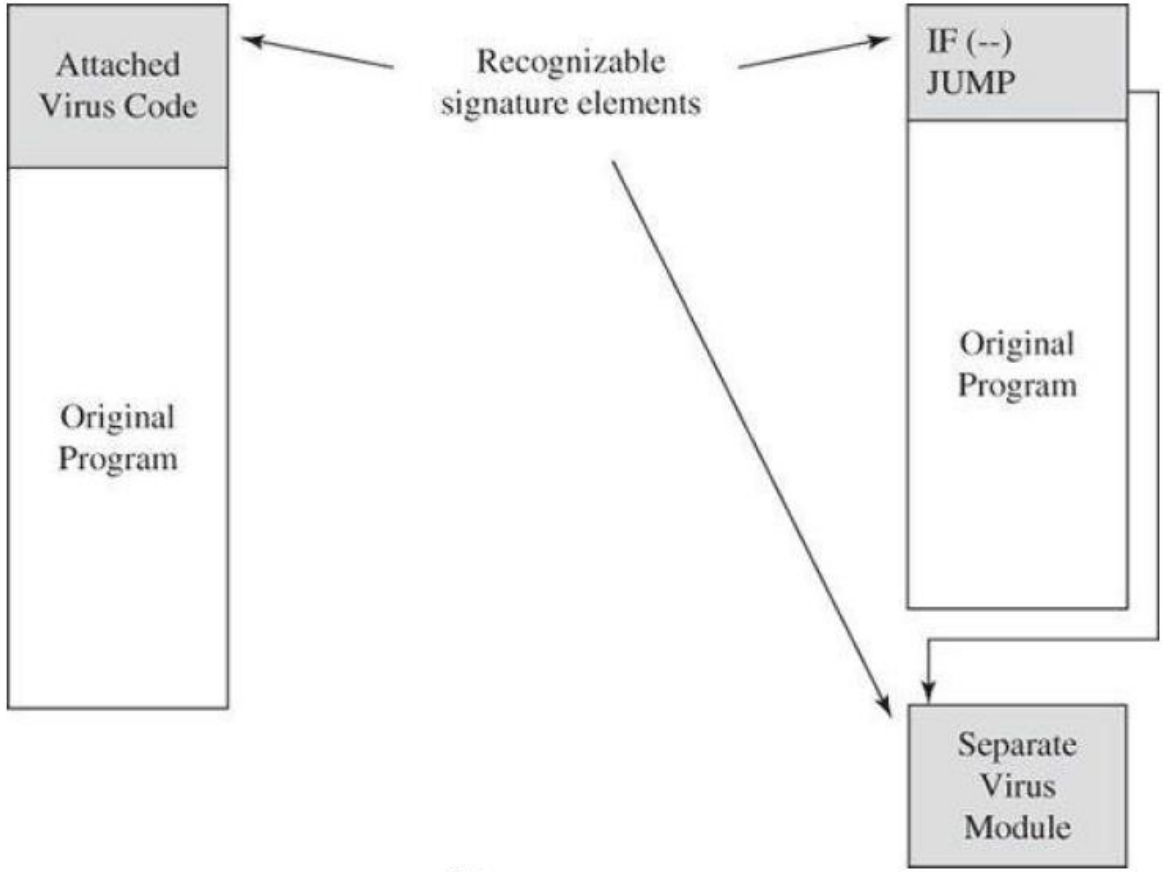


The Compilation Process: (a) Compilation. (b) Decompilation

Storage Patterns

Most viruses attach to programs that are stored on media such as disks. The attached virus piece is invariant, so 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.

Storage Patterns



Recognizable Patterns in Viruses

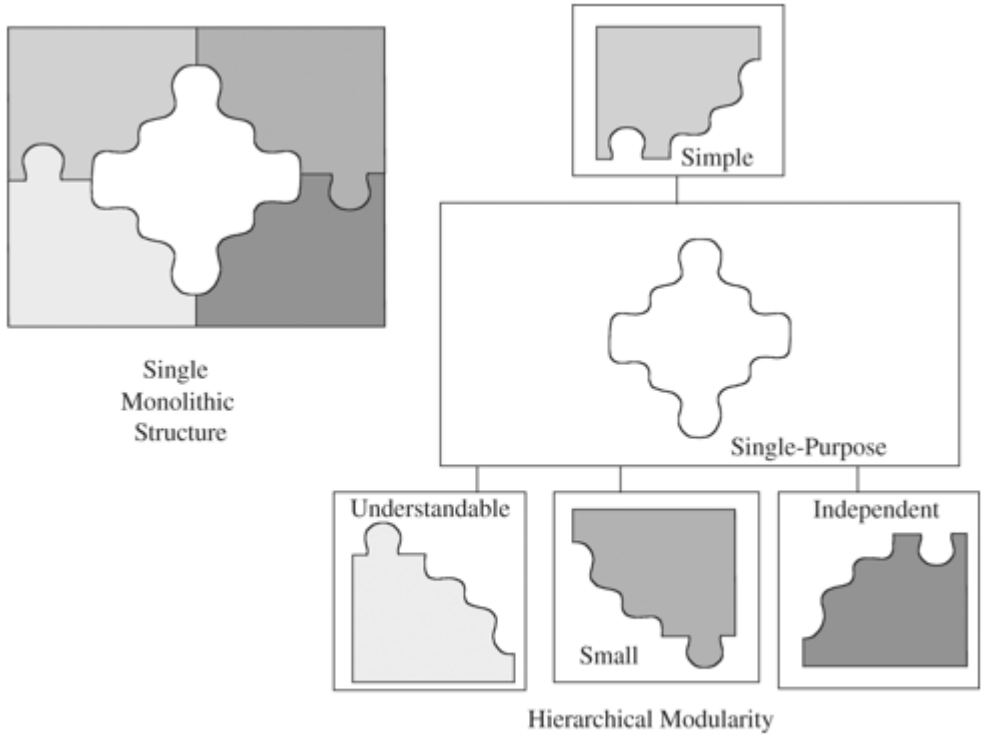
Countermeasures for Developers

Software engineering techniques:

- modular system
- encapsulation
- information hiding

Modularity

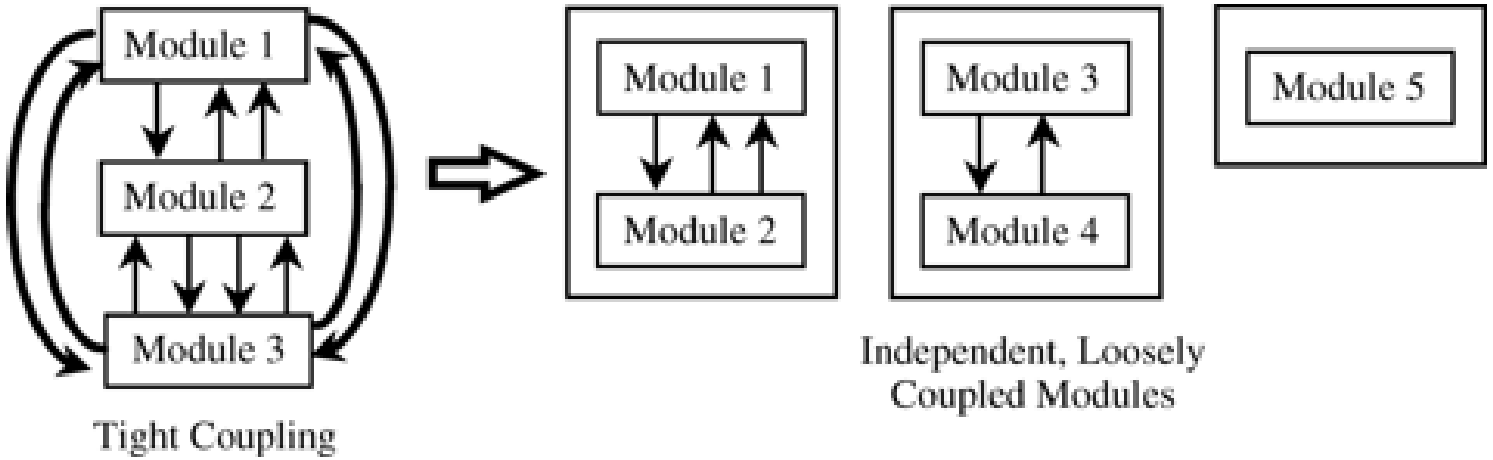
Modularization is the process of dividing a task into subtasks.



Modularity

Modularity

Coupling refers to the degree with which a component depends on other components in the system.



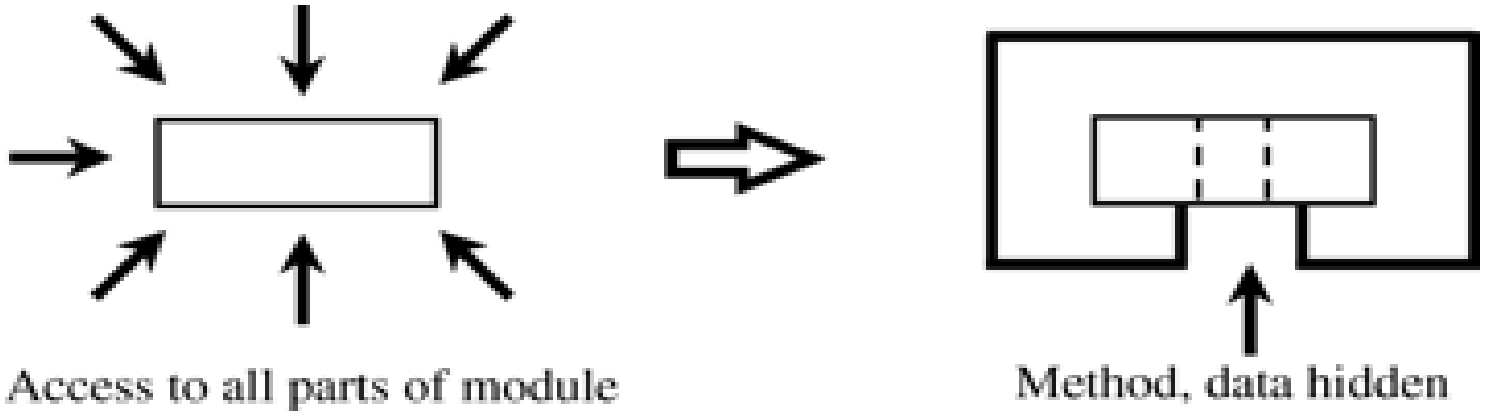
Coupling

Encapsulation

Encapsulation hides a component’s implementation details, but it does not necessarily mean complete isolation. Many computers must share information with other components. However, this sharing is carefully documented so that a component is affected only in known ways by other components in the system. Sharing is minimized so that the fewest interfaces possible are used.

Information Hiding

Information hiding: describing what a module does, not how.



Information Hiding

Testing

Testing is a process activity that concentrates on product quality: It seeks to locate potential product failures before they actually occur.

Testing

Types of testing:

- module testing
- component testing
- unit testing
- integration testing
- function test
- performance test
- acceptance test
- installation test