DENNIS GUNAWAN

IF470 COMPUTER SECURITY

04 MALICIOUS CODE

REVIEW: PROGRAM SECURITY

- Threat
  - Program Flaw Leads to Security Failing
- Vulnerability
  - Incomplete Mediation
  - Race Condition
  - Time-of-Check to Time-of-Use
  - Undocumented Access Point
- Ineffective Countermeasure
  - Penetrate-and-Patch
- Countermeasure
  - Identifying and Classifying Faults
  - Secure Software Design Elements
  - Secure Software Development Process
  - Testing
  - Defensive Programming

COURSE SUB LEARNING OUTCOMES (SUB-CLO)

- Sub-CLO 4
  - Students are able to relate malicious code to real case in their daily life (C3)

OUTLINE

- Threat
  - Malware – Virus, Trojan Horse, and Worm
- Technical Details
  - Malicious Code
- Vulnerability
  - Voluntary Introduction
  - Unlimited Privilege
  - Stealthy Behavior – Hard to Detect and Characterize
- Countermeasure
  - Hygiene
  - Detection Tools
  - Error Detecting and Error Correcting Codes
  - Memory Separation
  - Basic Security Principles

INTRODUCTION

Malicious Code

Programs such as viruses, worms, and Trojan horses that have been implanted in various applications

- Vex, confuse, or disrupt a user
- Expose or steal confidential information
- Destroy data or capability

INTRODUCTION

- Malicious code is deceptive
  - It can arrive without notice
  - It can execute just as stealthily
  - Its impact can be devastating and completely unpredictable

- Predictability is not a common characteristic of malicious code
  - Unpredictability makes preventing, detecting, confirming, and eradicating a code infection really difficult

INTRODUCTION

The range of malicious code activity

is practically unlimited

as new authors invent new code all the time

MALWARE

Malicious Code = Rogue Programs = Malware (Malicious Software)

General name for programs or program parts
planted by an agent with malicious intent
to cause unanticipated or undesired effects

MALWARE – VIRUS

Virus

A program that can replicate itself and pass on malicious code to other nonmalicious programs by modifying them

- The infected good program itself begins to act as a virus, infecting other programs

We cannot assume that
a clean program yesterday is still clean today

MALWARE – VIRUS

Transient Virus

- The lifespan depends on the life of its host
- The virus runs when the program to which it is attached executes
- It terminates when the attached program ends

Resident Virus

- Locates itself in memory
- It can remain active or be activated as a stand-alone program, even after its attached program ends

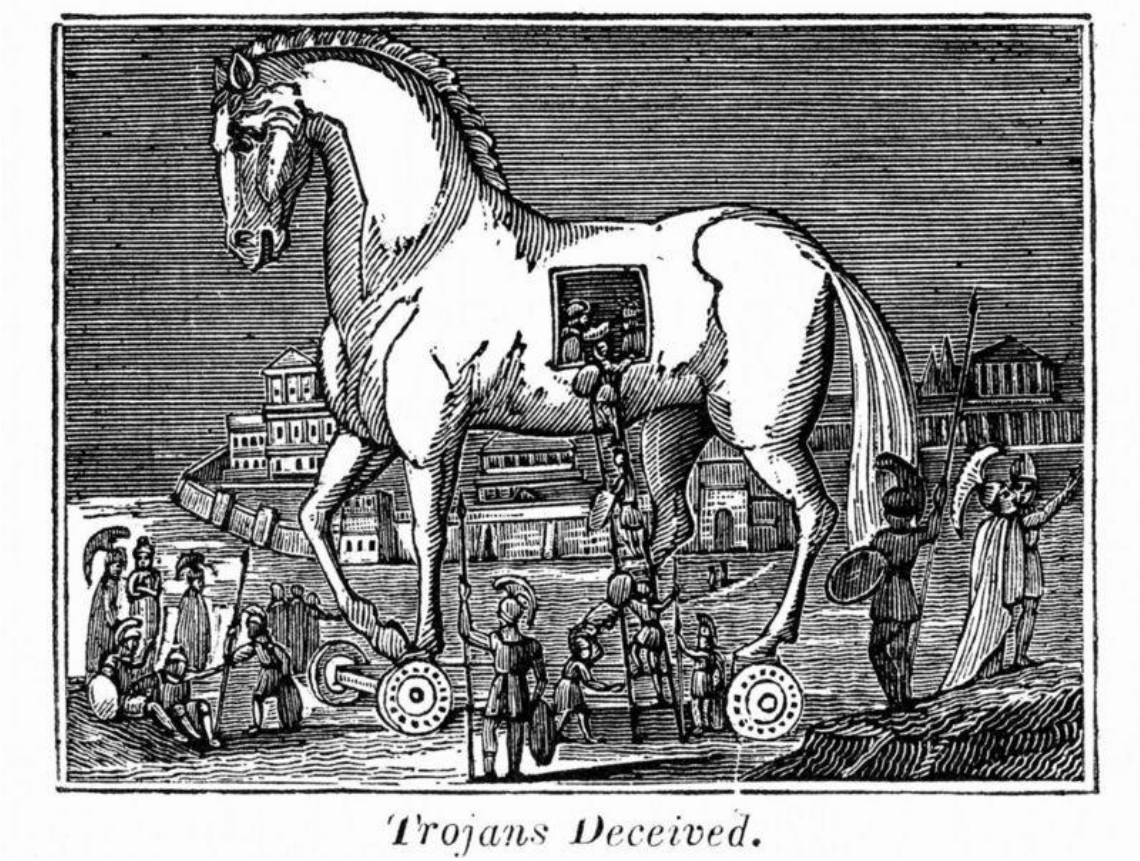
MALWARE – WORM

Worm

A program that spreads copies of itself through a network

Worm	Virus
Operates through networks	Can spread through any medium (but usually uses a copied program or data files)
Spreads copies of itself as a stand-alone program	Sometimes spreads copies of itself as a program that attaches to or embeds in other programs

MALWARE – TROJAN HORSE



Trojans Deceived.

Trojan Horse

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

MALWARE

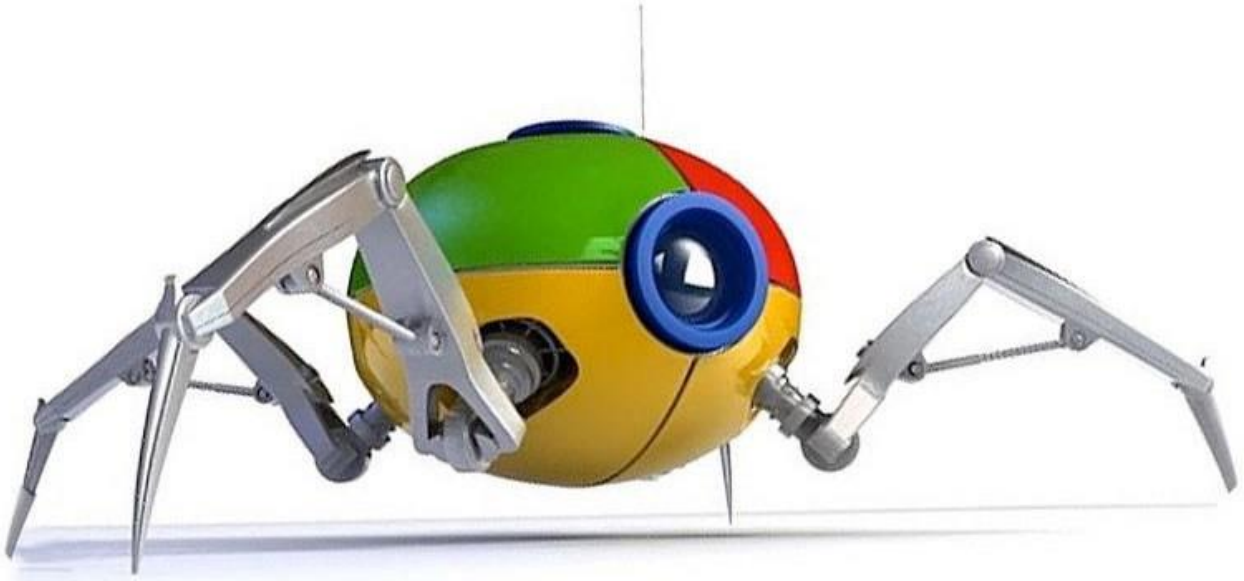
Code Type	Characteristics
Virus	Code that causes malicious behavior and propagates copies of itself to other programs
Trojan horse	Code that contains unexpected, undocumented, additional functionality
Worm	Code that propagates copies of itself through a network; impact is usually degraded performance
Rabbit	Code that replicates itself without limit to exhaust resources
Logic bomb	Code that triggers action when condition occurs
Time bomb	Code that triggers action when specified time occurs
Dropper	Transfer agent code only to drop other malicious code, such as a virus or Trojan horse
Hostile mobile code agent	Code communicated semiautonomously by programs transmitted through the web
Script attack, JavaScript, active code attack	Malicious code communicated in JavaScript, ActiveX, or another scripting language, downloaded as part of displaying a web page

MALWARE

Code Type	Characteristics
Bot	Semiautonomous agent, under control of a (usually remote) controller or "herder"; not necessarily malicious
Zombie	Code or entire computer under control of a (usually remote) program
Browser hijacker	Code that changes browser settings, disallows access to certain sites, or redirects browser to others
Rootkit	Code installed in "root" or most privileged section of operating system; hard to detect
Trapdoor or backdoor	Code feature that allows unauthorized access to machine or program; bypasses normal access control and authentication
Tool or toolkit	Program used to assemble (tool) a set of malicious code; not dangerous itself, but output (toolkit) intended for malice
Scareware	Not code; false warning of malicious code attack

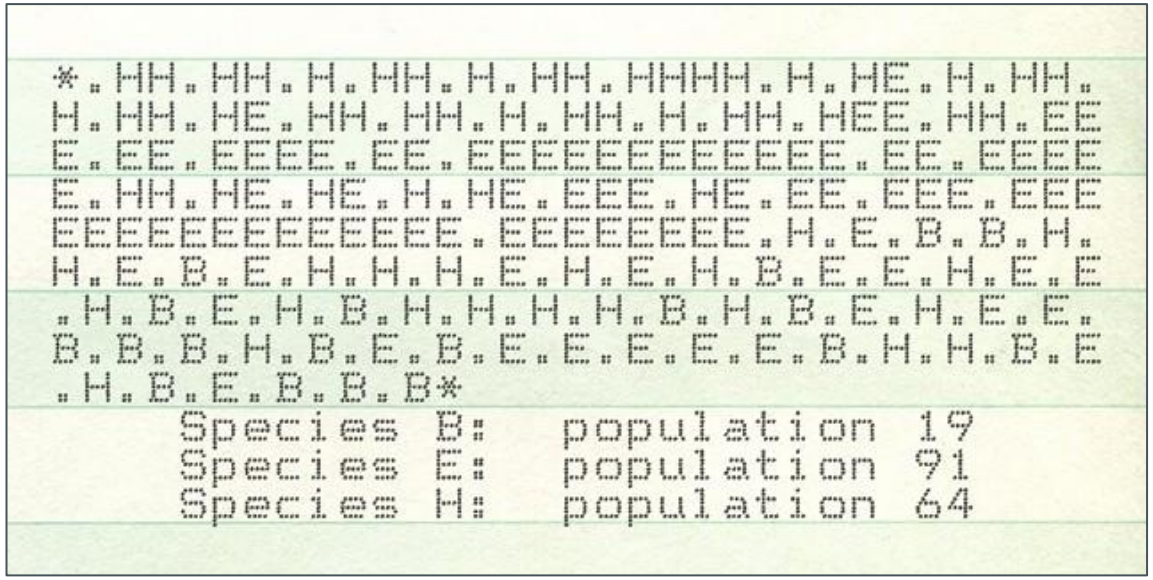
MALWARE

Not all activities listed here are always malicious



HISTORY OF MALICIOUS CODE

- Game Darwin
  - Developed by Vic Vyssotsky, Doug McIlroy, and Robert Morris of AT&T Bell Labs in 1962
  - Objective: kill opponents' programs
  - A battle among computer programs
    - The ability to reproduce and propagate
    - Hide to evade detection and extermination



HISTORY OF MALICIOUS CODE

- Through the 1980s and early 1990s
  - Malicious code was communicated largely person-to-person
    - Infected media (removable disks)
    - Documents transmitted through email (macros attached to documents)
  - Morris worm spread through the ARPANET (young and small Internet)
- Late 1990s
  - The use of Internet for communicating malicious code

HISTORY OF MALICIOUS CODE

- Malware continue to become more sophisticated
  - Exploitation of known system vulnerabilities
- Today's malware often stays dormant until needed, or until it targets specific types of software to debilitate some larger (sometimes hardware) system
  - Conficker (2008) leaves its targets under the control of a master agent
  - Stuxnet (2010) exploits a vulnerability in Siemens' industrial control systems software

HISTORY OF MALICIOUS CODE

Year	Name	Characteristics
1982	Elk Cloner	First virus, target is Apple II computers
1985	Brain	First virus to attack IBM PC
1988	Morris worm	Allegedly accidental infection disabled large portion of the ARPANET, precursor to today's Internet
1989	Ghostballs	First multipartite (has more than one executable piece) virus
1990	Chameleon	First polymorphic (changes form to avoid detection) virus
1995	Concept	First virus spread via Microsoft Word document macro

HISTORY OF MALICIOUS CODE

Year	Name	Characteristics
1998	Back Orifice	Tool allows remote execution and monitoring of infected computer
1999	Melissa	Virus spread through email address book
2001	Code Red	Virus propagates from 1st to 20th of month, attacks whitehouse.gov web site from 20th to 28th , rests until end of month, and restarts at beginning of next month; resides only in memory, making it undetected by file-searching antivirus products
2001	Code Red II	Like Code Red, but also installs code to permit remote access to compromised machines
2001	Nimda	Exploits known vulnerabilities; reported to have spread through 2 million machines in a 24-hour period
2003	Slammer worm	Attacks SQL database servers; has unintended denial-of-service impact due to massive amount of traffic it generates

HISTORY OF MALICIOUS CODE

Year	Name	Characteristics
2003	SoBig worm	Worm propagates by sending itself to all email addresses it finds; can fake From: field; can retrieve stored passwords
2004	MyDoom worm	Mass-mailing worm with remote-access capability
2004	Bagle or Beagle worm	Gathers email addresses to be used for subsequent spam mailings; SoBig, MyDoom, and Bagle seemed to enter a war to determine who could capture the most email addresses
2008	Rustock.C	Spam bot and rootkit virus
2008	Conficker	Virus believed to have infected as many as 10 million machines; has gone through five major code versions
2010	Stuxnet	Worm attacks SCADA (supervisory control and data acquisition) automated processing systems; zero-day attack

ZERO-DAY ATTACK

Use of malware

that exploits a previously unknown vulnerability or a known vulnerability

for which no countermeasure has yet been distributed

4 ASPECTS OF MALICIOUS CODE INFECTIONS

- Harm
  - How they affect users and systems
- Transmission and Propagation
  - How they are transmitted, replicate, and 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

Nondestructive

Often simply to show the author's capability

- Sending a funny message
- Flashing an image on the screen

Destructive

No apparent motive other than to harm the recipient

- Corrupt files
- Delete files
- Damage software
- Execute commands to cause hardware stress or breakage

HARM FROM MALICIOUS CODE

Commercial or Criminal Intent

- Overtake the recipient's computer
- Installing code to allow a remote agent
  - Cause the computer to perform actions on the agent's signal
  - Forward sensitive data to the agent

- Collecting personal data
  - Login credentials to a banking web site
- Collecting proprietary data
  - Corporate plans
    - An infection of computers of 5 petroleum industry companies in February 2011
- Serving as a compromised agent for sending spam email or mounting a denial-of-service attack

Virus Hoaxes

Messages falsely warning of a piece of malicious code,
apparently to cause receivers to panic and
forward the message to contacts, thus spreading the panic

HARM TO THE USER

- 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
- Modifying system program files
- Modifying system information
- Stealing and forwarding sensitive information
- The user can be harmed indirectly

HARM TO THE USER'S SYSTEM

- 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
- 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 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 disabled

HARM TO THE WORLD

- Computer users and society in general bear a heavy cost for dealing with malware
  - Morris worm of 1988 infected only 3,000 computers
  - ILOVEYOU worm of 2000 is estimated to have infected 100,000 servers
    - 1 email of every 28 transmitted was an infection from the worm
  - Code Red is believed to have affected close to 3 million hosts
    - Estimate of damage from Code Red range from $500 million to $2.6 billion
  - One estimate of the damage from Conficker, for which 9 to 15 million systems were repaired (plus 1.5 million not yet cleaned of the infection), was $9.2 billion, or roughly $1,000 per system

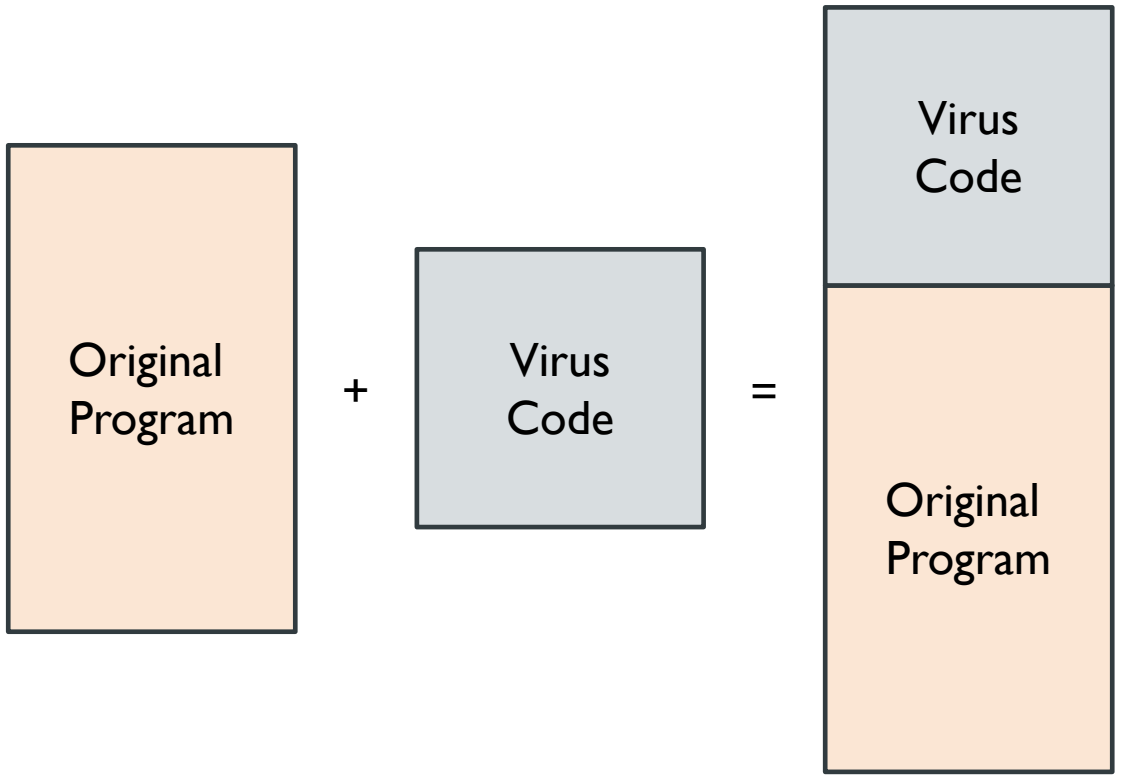
TRANSMISSION AND PROPAGATION

- Setup and installer program transmission
- Attached file
- Download from web site
- Document viruses
- Malicious scripts
- Autorun
- Distribution via flash memory
- Drive-by downloads

- Some modern email handlers, in a drive to "help" the receiver (victim), automatically open attachments as soon as the receiver opens the body of the email message
- Handing out free drives as advertising at a railway station

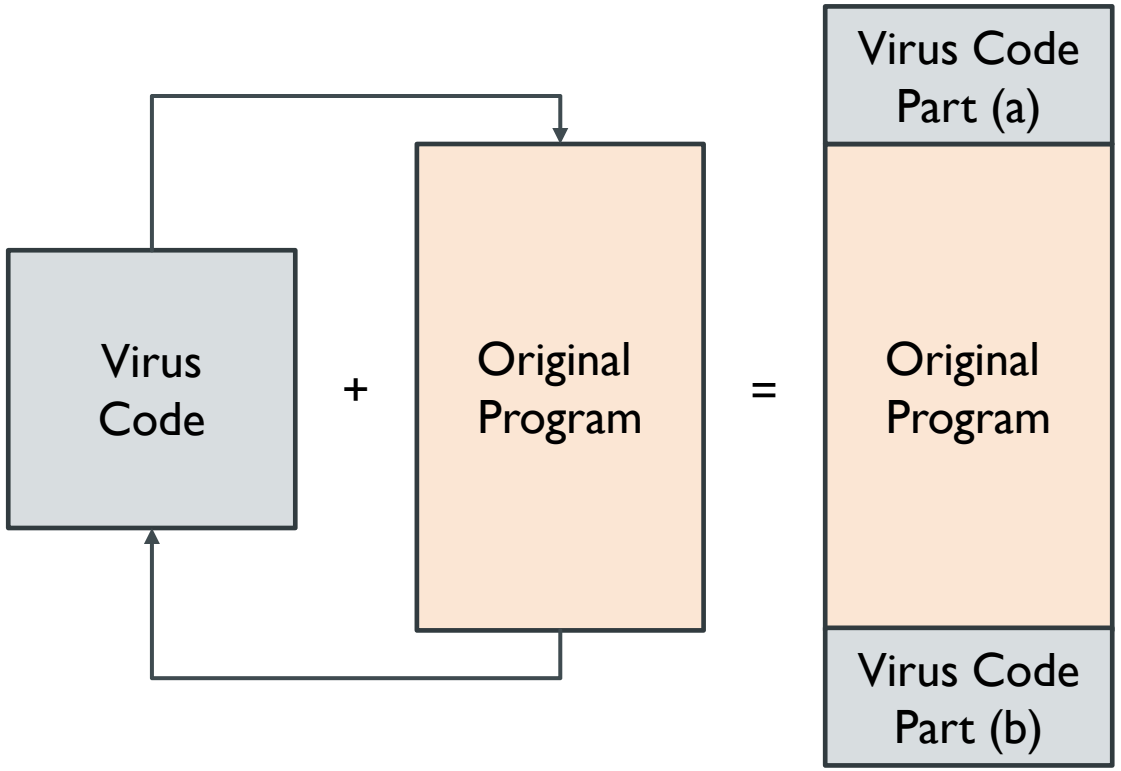
HOW VIRUSES ATTACH – APPENDED VIRUSES

- A virus inserts a copy of itself into the executable program file before the first executable instruction
- The virus performs its task and then transfers to the original program
- The virus writer need not know anything about the program to which the virus will attach



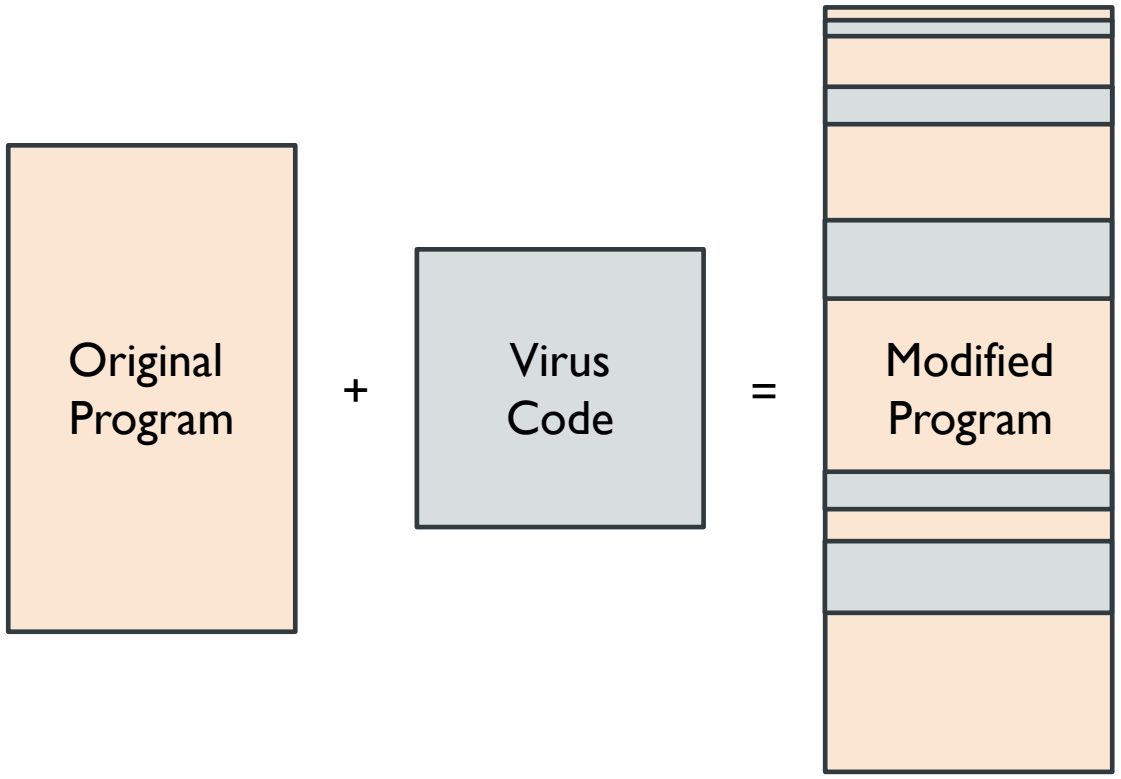
HOW VIRUSES ATTACH – VIRUSES THAT SURROUND A PROGRAM

- A virus runs the original program but has control before and after execution



HOW VIRUSES ATTACH – INTEGRATED VIRUSES AND REPLACEMENTS

- 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



ACTIVATION

- Early malware writers used document macros and scripts as the vector for introducing malware into an environment
- Malware now often exploits one or more existing vulnerabilities in a commonly used program
  - Hope the intended victim has not yet applied a protective or corrective patch

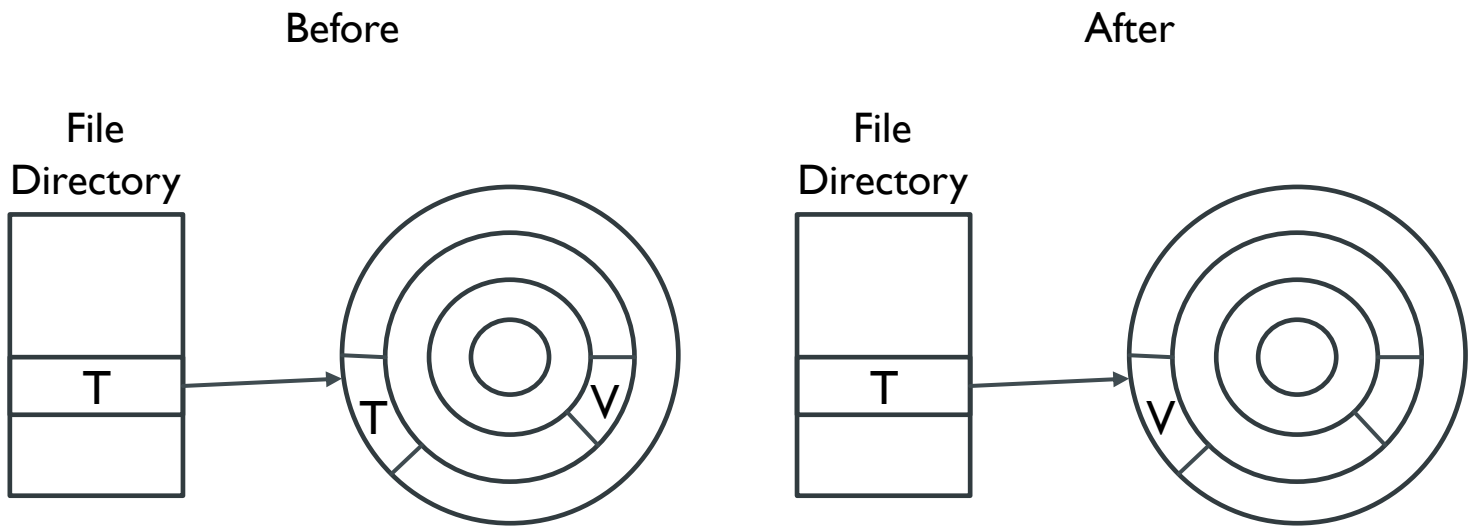
- Having discovered a flaw, a security researcher or a commercial software vendor faces a dilemma
  - Announce the flaw for which there may not yet be a patch
  - Keep quiet and hope the malicious code writers have not yet discovered the flaw

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)
  - The virus has to seem to be T
    - Saying effectively, "I am T"
  - The virus has to push T out of the way and become a substitute for T
    - Saying effectively, "Call me instead of T"

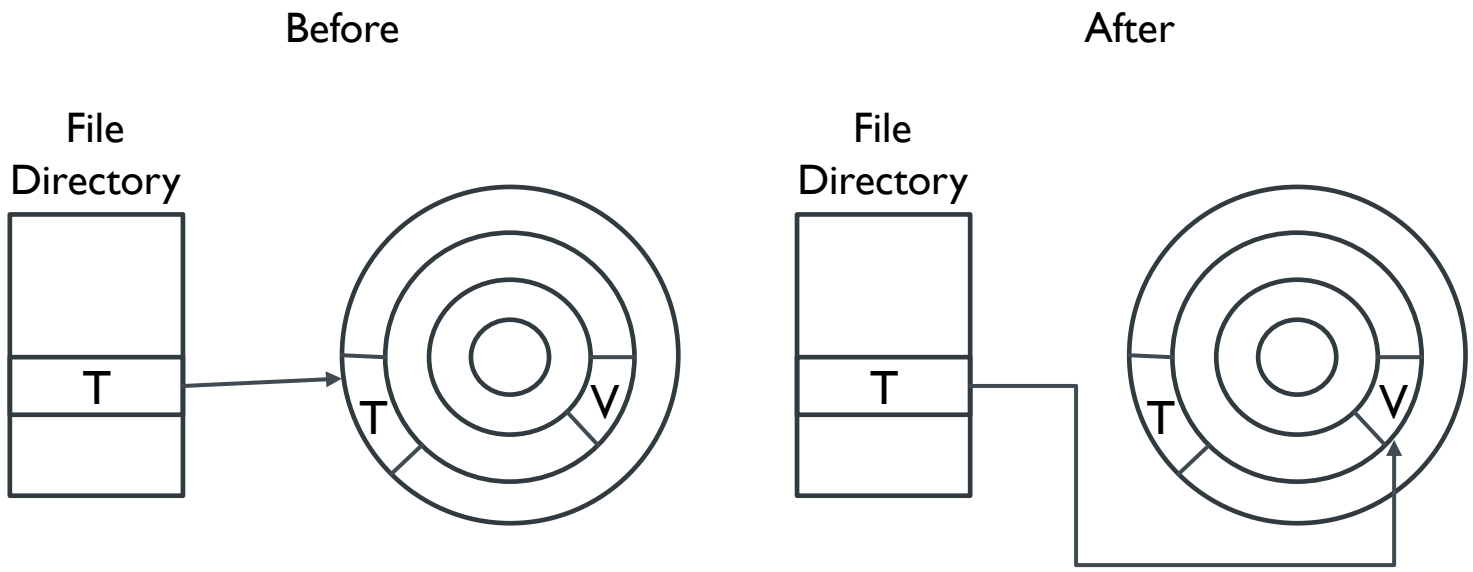
HOW MALICIOUS CODE GAINS CONTROL

- The virus can overwrite T in storage (simply replacing the copy of T in storage, for example)



HOW MALICIOUS CODE GAINS CONTROL

- The virus can change the pointers in the file table so that the virus is located instead of T whenever T is accessed through the file system



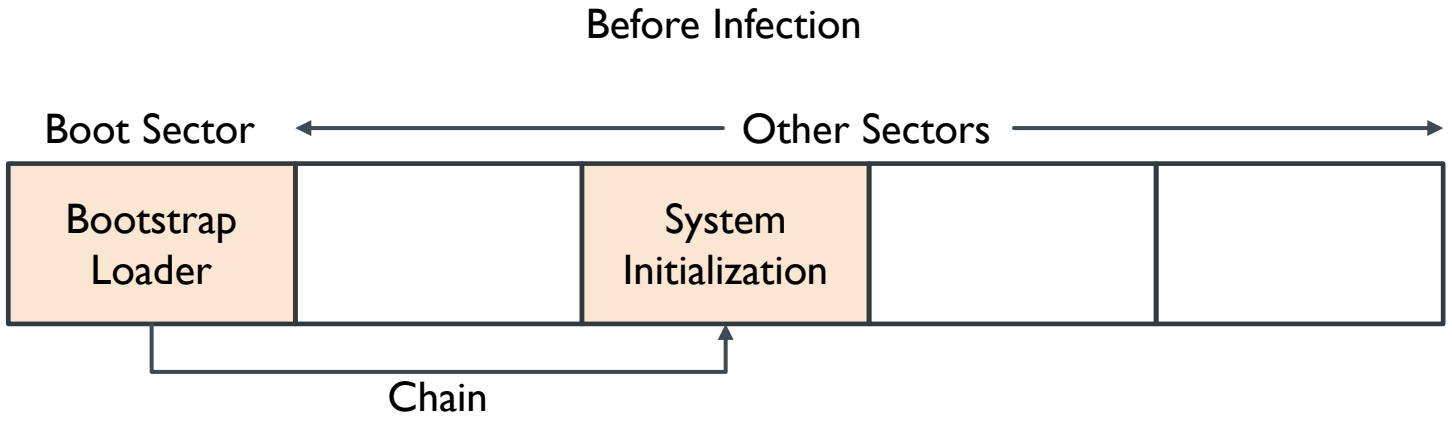
EMBEDDING: HOMES FOR MALWARE

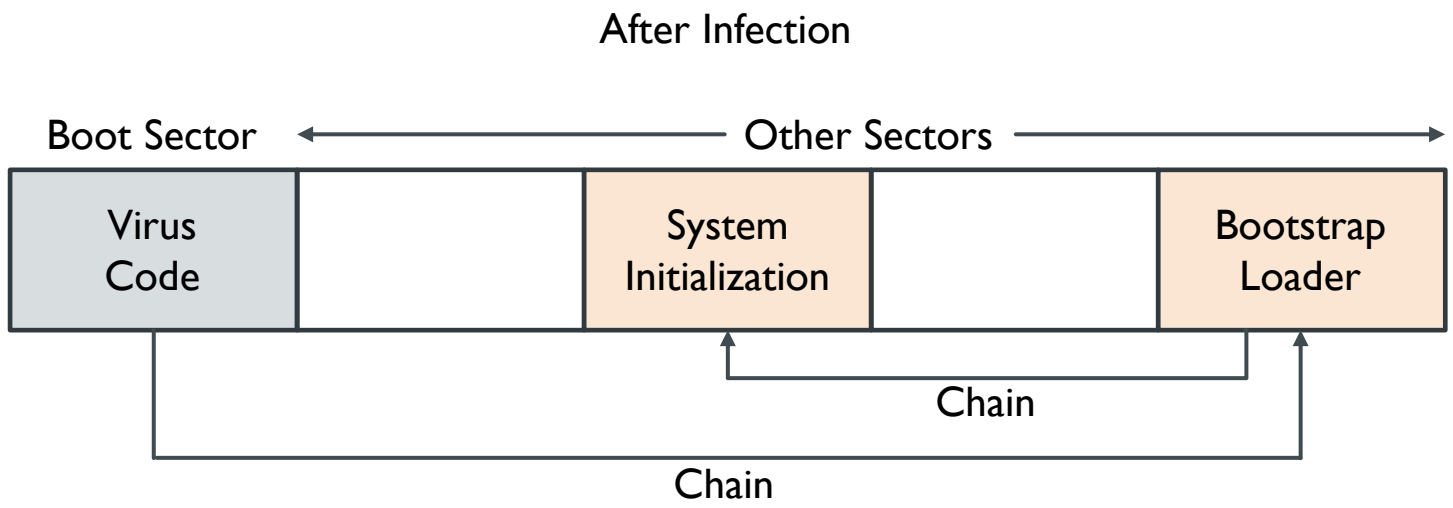
- The malware writer may find it appealing to build these qualities into the malware
  - Hard to detect
  - Not easily destroyed or deactivated
  - Spreads infection widely
  - Can reinfect its home program or other programs
  - Easy to create
  - Machine independent and operating system independent

ONE-TIME EXECUTION (IMPLANTING)

- Malicious code often executes a one-time process to transmit or receive and install the infection
  - The user clicks to download a file
  - The user opens an attachment
  - The malicious code is downloaded silently as a web page is displayed
- This first step to acquire and install the code must be quick and not obvious to the user

BOOT SECTOR VIRUSES





- The virus gains control very early in the boot process, before most detection tools are active
- The virus code is not readily noticed by users
  - The files in the boot area are crucial parts of the operating system
  - The operating system makes them "invisible"

MEMORY-RESIDENT VIRUSES

"Resident" Code

Code that remains in memory

Example:
routine that interprets keys pressed on the keyboard

- Virus writers also like to attach viruses to resident code
  - The resident code is activated many times while the machine is running
  - Each time the resident code runs, the virus does too
  - Once activated, the virus can look for and infect uninfected carriers

OTHER HOMES FOR VIRUSES

- Application program
  - "Macro" feature
- Libraries
  - Used by many programs
  - Shared among users and transmitted from one user to another
- Compilers, loaders, linkers, runtime monitors, runtime debuggers, and even virus control programs
  - Widely shared

STEALTH

- Installation Stealth
  - Transmit code without the user being aware
  - Tricking the user into accepting malware
- Execution Stealth
  - Remaining unnoticed during execution
  - Modern operating systems often support dozens of concurrent processes, many of which have unrecognizable names and functions

STEALTH

- 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
      - A := 1; B := 2 can be rearranged with no detrimental effect
    - Insert instructions that have no result
      - A := A
    - Insert random strings
      - Perhaps as constants that are never used
    - Insert instructions that are never executed
      - In the else part of a conditional expression that is always true
    - Replace instructions with others of equivalent effect
      - Replacing A := B – 1 with A := B + (-1)

VOLUNTARY INTRODUCTION

- The easiest way for malicious code to gain access to a system is to be introduced by a user, a system owner, an administrator, or other authorized agent

- Because you cannot always know which sources are infected, you should assume that any outside source in infected
  - It is not feasible to cut off all contact with the outside world
  - Malware seldom comes with a big warning sign
  - Malware is often designed to fool the unsuspecting

UNLIMITED PRIVILEGE

- Many home computers are designed for one user who is all-powerful
  - Administration is simplest if the user has full privileges on the computer
    - What if an update is needed or new software requires installation?
  - If the user inadvertently activates a piece of malicious code, that code runs with unlimited user privileges
- A more prudent organization recognizes two or more user identities
  - Administrator and ordinary user

STEALTHY BEHAVIOR – HARD TO DETECT AND CHARACTERIZE

- Malicious code can attach itself to benign programs
- Writers of malware vary their code's effect
  - Behavior is inconsistent
- Unpredictability adds to the ability of malicious behavior to remain undetected

HYGIENE

Not engaging in behavior that permits malicious code contamination

- 2 components
  - Avoiding points of contamination
  - Blocking avenues of vulnerability

HYGIENE

- Several techniques for building a reasonably safe community for electronic contact
  - 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
  - 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

HYGIENE

- As new vulnerabilities become known, you should apply patches

- Problems
  - A vendor's patch cannot and does not consider possible interactions with other software
  - A patch to one software application has been "recognized" incorrectly by an antivirus checker to be malicious code

HYGIENE

Good hygiene and self-defense

are important controls against malicious code

DETECTION TOOLS

Virus Scanners

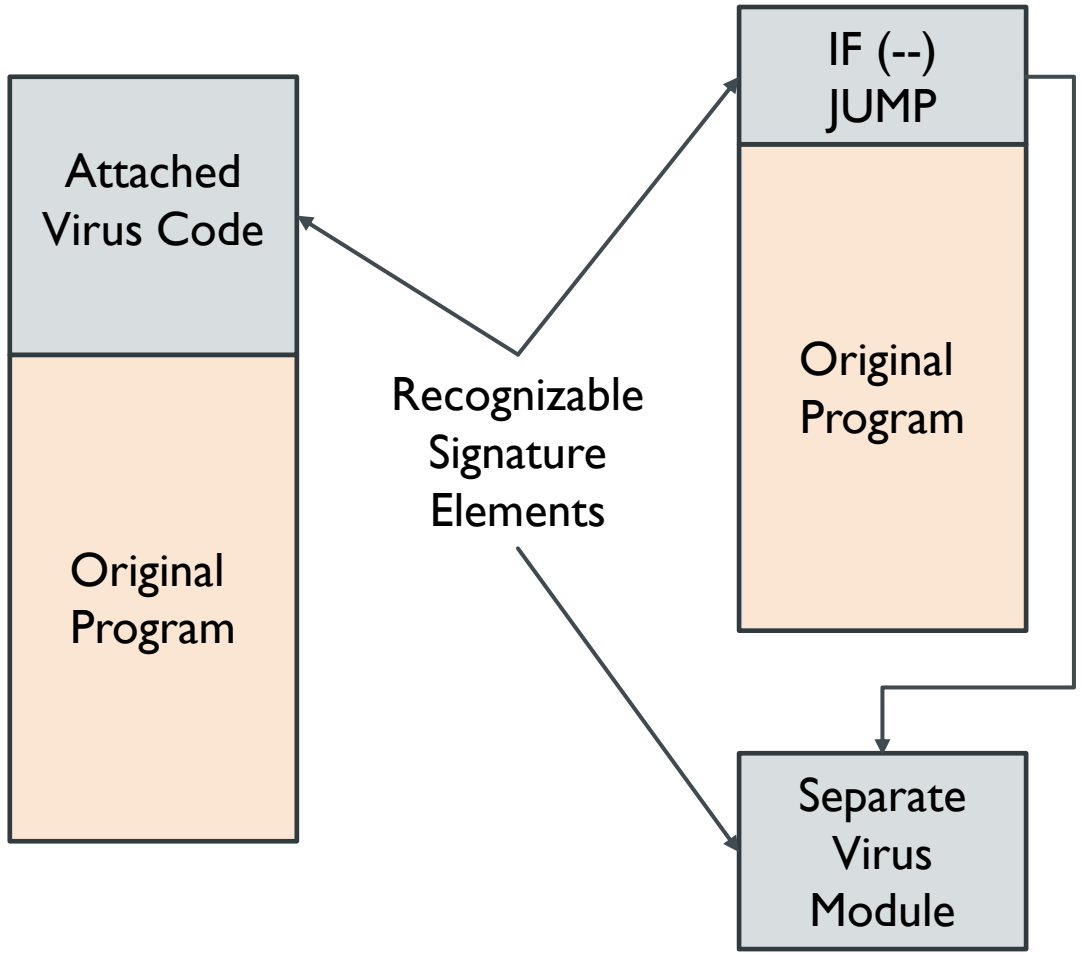
Tools that look for signs of malicious code infection

- Look for signatures or patterns of viruses in memory and long-term storage
- A virus scanner is effective only if it has been kept up to date with the latest information on current viruses

DETECTION TOOLS

- A scanner looking for signs of the Code Red worm can look for a pattern containing the following characters

/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
%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



DETECTION TOOLS

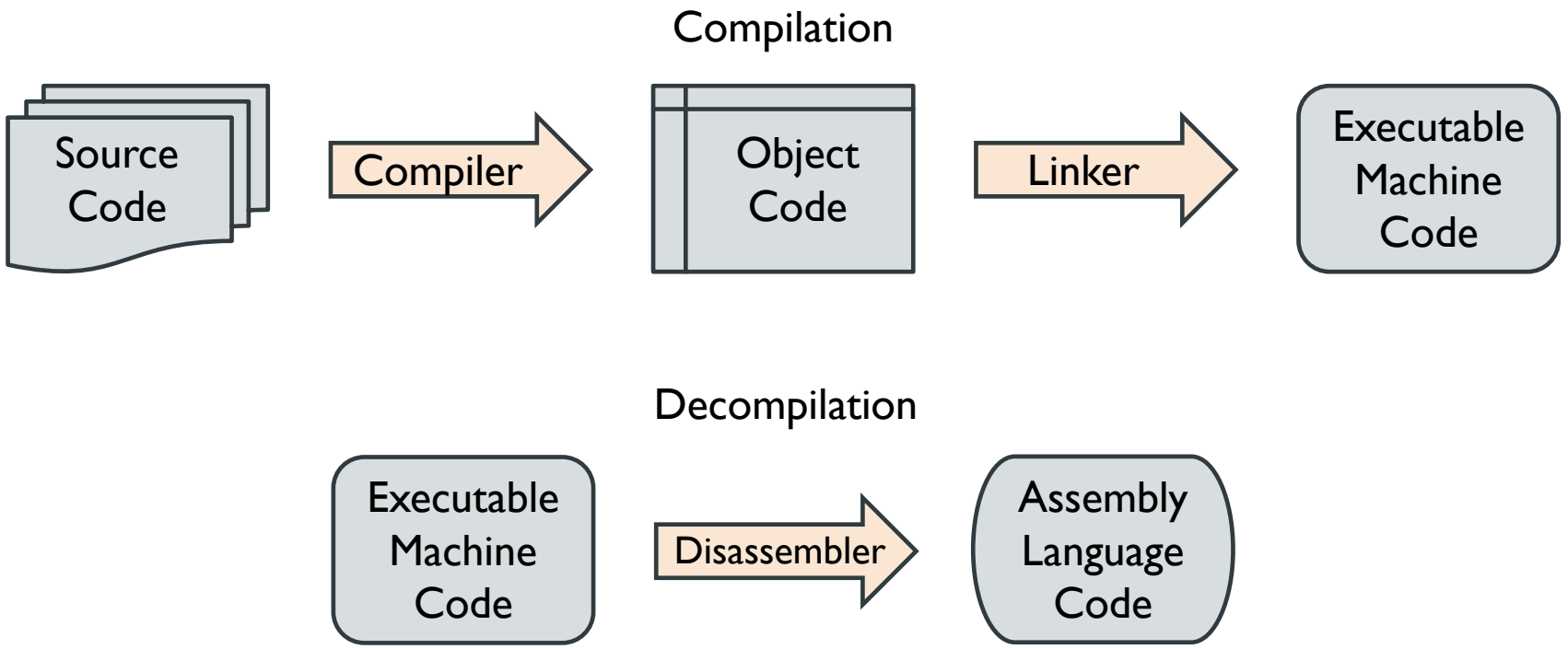
- 2 major limitations
  - Detection tools are necessarily retrospective
    - Looking for patterns of known infections
  - Patterns are necessarily static
    - Malicious code writers are conscious of pattern matching, so they vary their code to reduce the number and length of repeated patterns

CODE ANALYSIS

- One 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 first difficulty with analyzing code is that the researcher normally has only the end product to look at

CODE ANALYSIS



POLYMORPHIC VIRUS

A virus

that can change its appearance

- Instructions that cause no effect can be sprinkled into a piece of code to distort any pattern
- A polymorphic virus has to randomly reposition all parts of itself and randomly change all fixed data

ENCRYPTING VIRUS

A simple variety of polymorphic virus that uses different encryptions under various keys to make the stored form of the virus different

- 3 distinct parts
  - A decryption parameter
  - The (encrypted) object code of the virus
  - The (unencrypted) object code of the decryption routine
- The decryption routine itself or a call to a decryption library routine must be in the clear
  - Becomes the signature

FAKE ANTIVIRUS TOOLS

- Provide malware, with effects ranging from locking up a computer to demanding money to clean up nonexistent infections

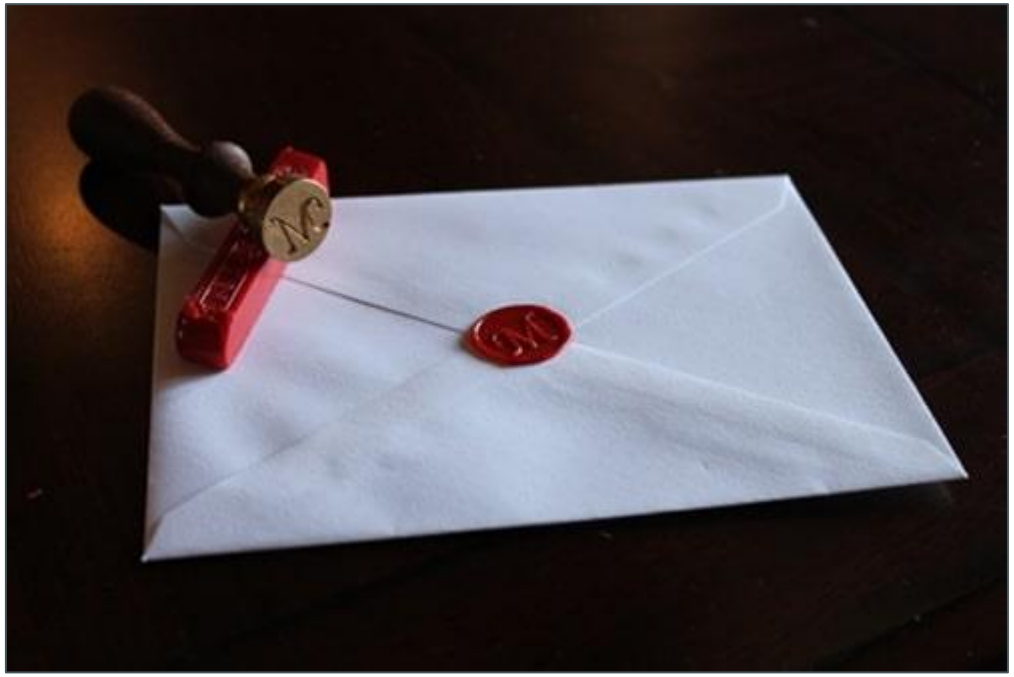
- Be careful acquiring software from unknown sources

ERROR DETECTING AND ERROR CORRECTING CODES

Guard against modification of data

- Error detecting codes detect when an error has occurred
- Error correcting codes can actually correct errors without requiring a copy of the original data

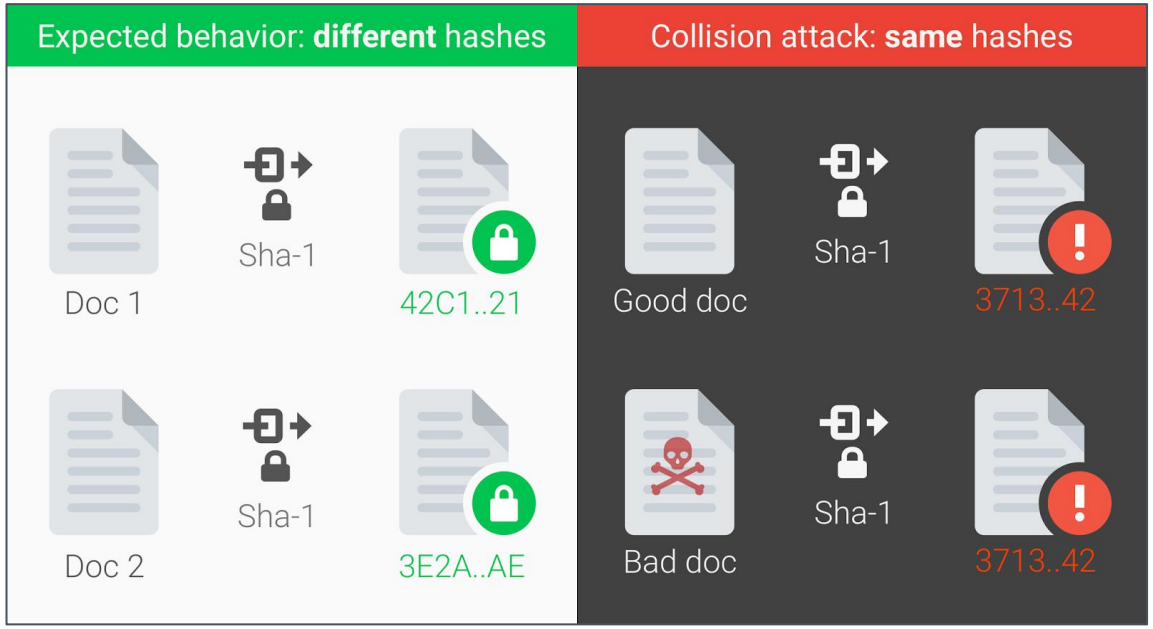
HASH CODES / CRYPTOGRAPHIC CHECKSUM



- Hash = checksum = message digest
- One-way functions
  - Functions which are much easier to compute than their inverses
    - $y = x^3$
      - $\sqrt[3]{y}$ is much more difficult to compute
    - $y = x^2$
      - There are 2 possibilities for $\sqrt[2]{y}$: $+x$ and $-x$

HASH CODES / CRYPTOGRAPHIC CHECKSUM

- The most widely used cryptographic hash functions
  - MD4 & MD5
    - Both condense a message of any size to a 128-bit digest
  - SHA / SHS (Secure Hash Algorithm or Standard)
    - Produces a 160-bit digest



TRIPWIRE

The most well-known software that compares the active version of a software code with a saved version of a digest of that code

MEMORY SEPARATION

Keeping one user's objects separate from other users

- Physical separation
- Temporal separation
- Logical separation
- Cryptographic separation

BASIC SECURITY PRINCIPLES

- Least Privilege
  - A subject should have access to the smallest number of objects necessary to perform some task
  - Separating user and administrative rights
  - Reduces the ability for malicious code to be implanted and bounds its impact if it should become installed
- Complete Mediation
  - Every potential access to an object must be checked
  - Guards against unacceptable access to objects
    - User data, programs, devices (USB flash memory), and services (email)

REFERENCES

- Pfleeger, Charles P. and Shari Lawrence Pfleeger (2012), *Analyzing Computer Security*, 1st Edition, Prentice Hall.

NEXT WEEK: ILLICIT DATA ACCESS

- Attack
  - Keylogging
- Threat
  - Illicit Data Access
- Harm
  - Data and Reputation
- Vulnerability
  - Physical Access
  - Misplaced Trust
  - Insiders
  - System Subversion
  - Weak Authentication
- Failed Countermeasure
  - Security through Obscurity
- Countermeasure
  - Physical Access Control
  - Strong Authentication
  - Trust / Least Privilege

AS THE WORLD IS INCREASINGLY INTERCONNECTED, EVERYONE SHARES THE RESPONSIBILITY OF SECURING CYBERSPACE

Vision

To become an **outstanding** undergraduate Computer Science program that produces **international-minded** graduates who are **competent** in software engineering and have **entrepreneurial spirit** and **noble character**.

INFORMATIKA
UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

Mission

1. To conduct studies with the best technology and curriculum, supported by professional lecturer
2. To conduct research in Informatics to promote science and technology
3. To deliver science-and-technology-based society services to implement science and technology