Chapter 15: System Security

- The slides do not contain all the information and cannot be treated as a study material for Operating System. Please refer the text book for exams.

Topics

- The security problem
- Program threats
- System and Network threats
- User Authentication

The Security Problem

- System **secure** if resources used and accessed as intended under all circumstances
  - Unachievable
- Intruders (crackers) attempt to breach security
- **Threat** is potential for security violation
- **Attack** is attempt to break security
- Attack can be accidental or malicious
- Easier to protect against accidental than malicious misuse

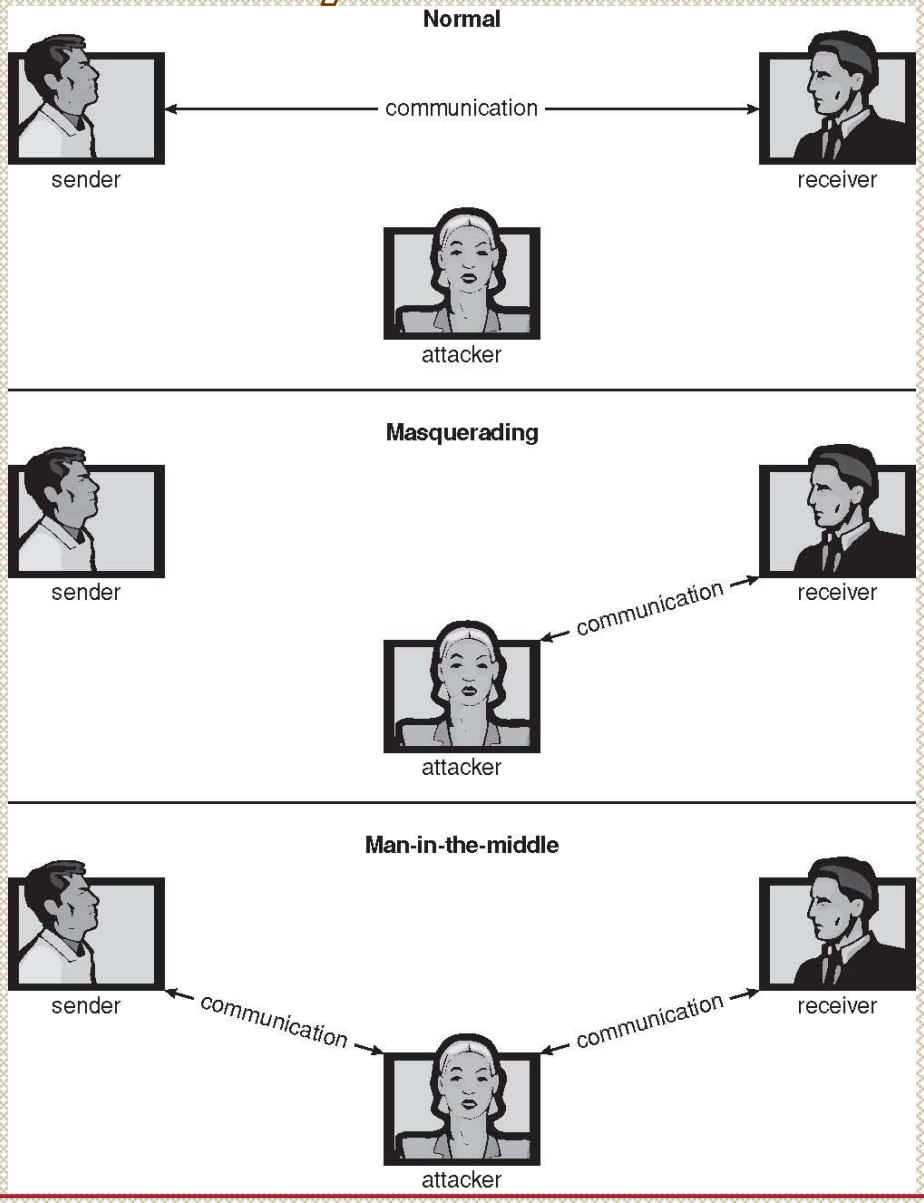
Security Violation Categories

- **Breach of confidentiality**
  - Unauthorized reading of data – credit card info or identity info
- **Breach of integrity**
  - Unauthorized modification of data – change source code
- **Breach of availability**
  - Unauthorized destruction of data – web-site defacement
- **Theft of service**
  - Unauthorized use of resources – install a daemon that acts as file server
- **Denial of service (DOS)**
  - Prevention of legitimate use of the system

Security Violation Methods

- **Masquerading** (breach **authentication**)
  - Pretending to be an authorized user to escalate privileges
- **Replay attack**
  - As is or with message modification – repeat the request to transfer money
- **Man-in-the-middle attack**
  - Intruder sits in data flow, masquerading as sender to receiver and vice versa
- **Session hijacking**
  - Intercept an already-established session to bypass authentication - preceded by man in the middle

Standard Security Attacks



Security Measure Levels

- Impossible to have absolute security, but make cost to perpetrator sufficiently high to deter most intruders
- Security must occur at four levels to be effective:
- **Physical**
  - Data centers, servers, connected terminals must be secured
- **Human**
  - Authorized users tricked into allowing access via **social engineering**.
  - **Phishing** – legitimate email makes user enter confidential information
  - **Dumpster diving** – general attempt to gather information to gain unauthorized access – look through books trash to find passwords

Security Measure Levels

- **Operating System**
  - process could constitute a denial of service attack, query to service could reveal passwords
- **Network**
  - Computer data travels over private leased lines, wireless connections – interruption of communication could result in DOS
- Security is as weak as the weakest link in the chain
- New hardware features are allowing systems to be made more secur.

Program Threats

1. **Trojan Horse**

- Code segment that misuses its environment
- Exploits mechanisms for allowing programs written by users to be executed by other users
- Deleting files in text editors using "." In the current search path.
- **Spyware –** Accompanies a program that the user chose to install – download ads to display,
- create **pop-up browser windows** when certain sites are visited
- Capture information from user's system and return it to central site - **covert channels**
- Up to 80% of spam delivered by spyware-infected systems

Program Threats

1. **Trojan Horse**

- User of OS does not need to install network daemons
- Such daemons are installed via 2 mistakes
- User run with more privileges than necessary(admin)
- OS may allow by default more privileges than a normal user needs – poor design

Program Threats

- **Trap Door**
- Designer leaves a hole in the software that only he is capable of using
- Bank rounding errors – occasional half cent credited to their accounts – adding up to large amount
- Pose difficult problem as we have to analyze all the source code – millions of lines of code

Program Threats

- **Logic Bomb**
- Program that initiates a security incident under certain circumstances
- Hard to detect – when a predefined set of parameters are met – detect if still employed else allow remote access

Program Threats

- **Stack and Buffer Overflow**
- Exploits a bug in a program (overflow either the stack or memory buffers) – the attacker determines the vulnerability and writes a program to do the following

1. Overflow an input field or command line argument until it writes into the stack
2. Overwrite the return address on the stack with the address of the exploit code loaded
3. Write a simple set of code for the next space in the stack that includes commands that the attacker wishes to execute – spawn a shell

- Unauthorized user or privilege escalation

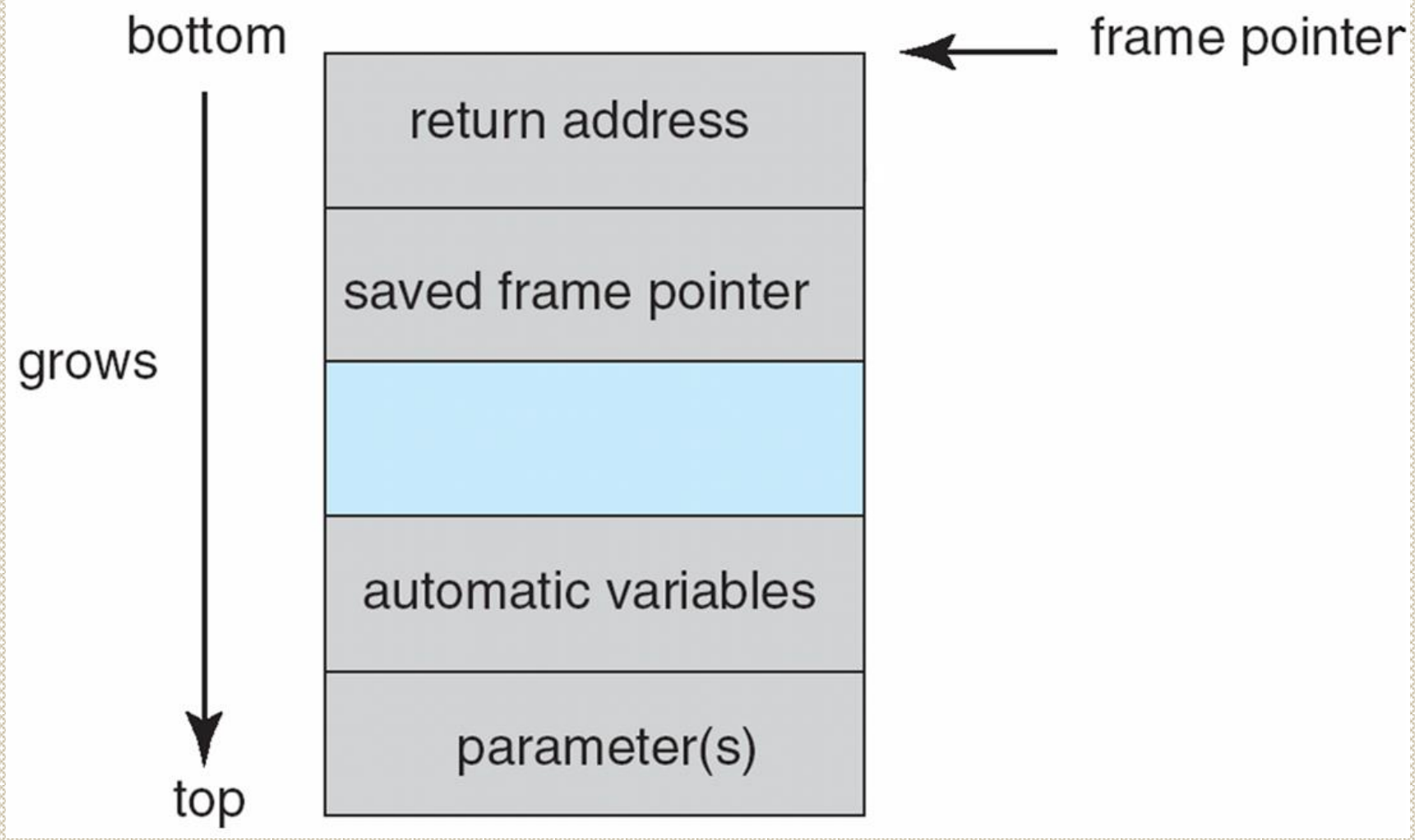
Program Threats

- Web page form expects a username – attacker could send user name +
  - extra characters to overflow the buffer and reach the stack,
  - plus a new address to load onto the stack,
  - plus the code
- When the buffer reading subroutine returns from execution, the return address is the exploit code , and the code is run
- Attack is harmful because it can be run between systems and can travel over allowed communication channels

C Program with Buffer-overflow Condition

```
#include <stdio.h>
#define BUFFER SIZE 256
int main(int argc, char *argv[])
{
  char buffer[BUFFER SIZE];
  if (argc < 2)
      return -1;
  else {
      strcpy(buffer,argv[1]);
      return 0;
  }
}
```

Layout of Typical Stack Frame

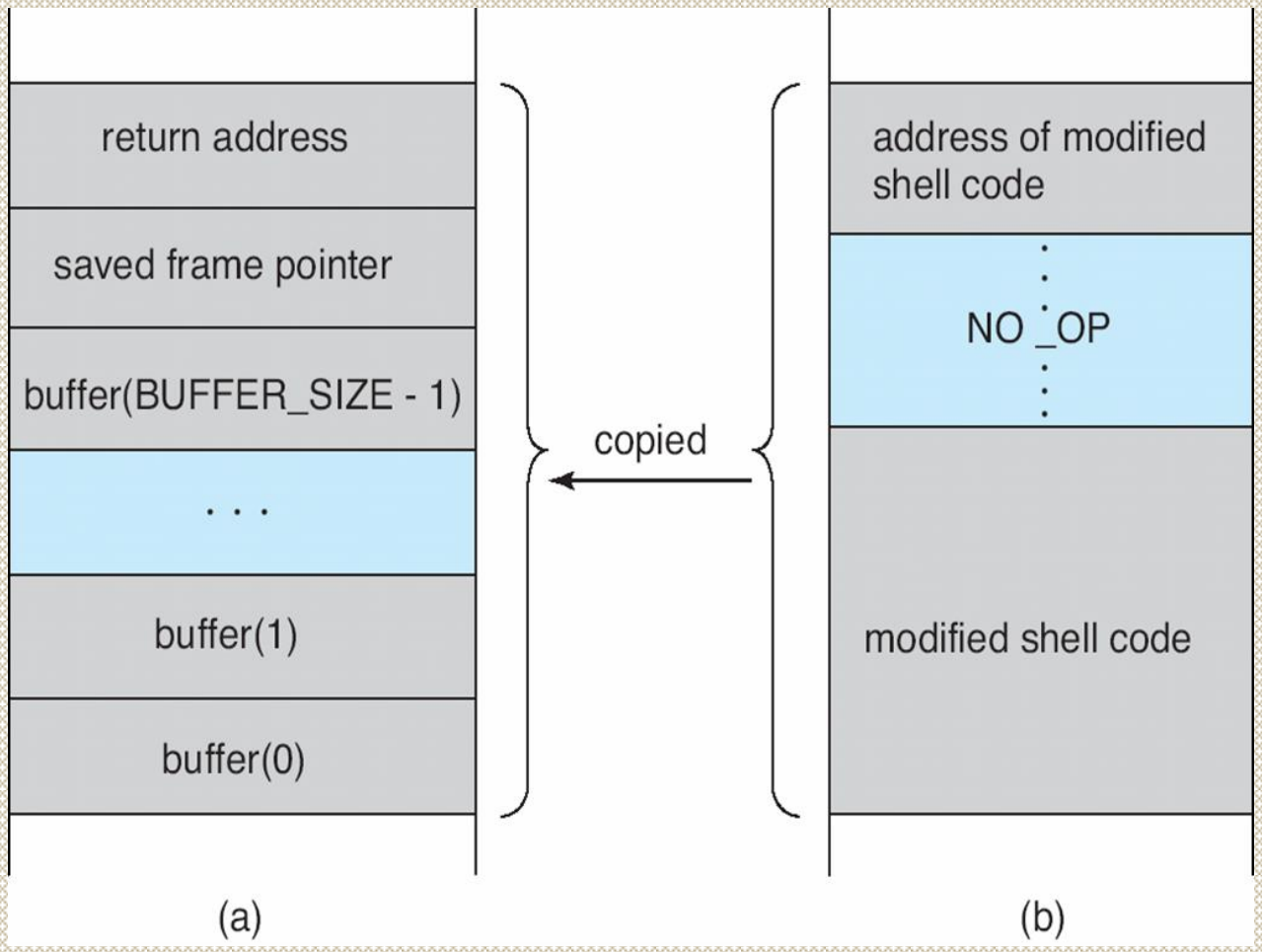


Modified Shell Code

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  execvp(``\bin\sh'',``\bin \sh'', NULL);
  return 0;
}
```

Hypothetical Stack Frame

(a) - Before
(b) - after



Great Programming Required?

- Program being attacked ran with system wide permission
- Considerable knowledge and programming skill required to recognize exploitable code and to exploit it
- Buffer overflow can be disabled by disabling stack execution or adding bit to page table to indicate “non-executable” state
  - Available in SPARC and x86

Program Threats (Cont.)

- **Viruses**
- Code fragment embedded in legitimate program
- Self-replicating, designed to infect other computers
- Very specific to CPU architecture, operating system, applications
- Usually borne via email or as a macro
  - Visual Basic Macro to reformat hard drive

```
Sub AutoOpen()
Dim oFS
  Set oFS =
  CreateObject(''Scripting.FileSystemObject'')
  vs = Shell(''c:command.com /k format c:'',vbHide)
End Sub
```

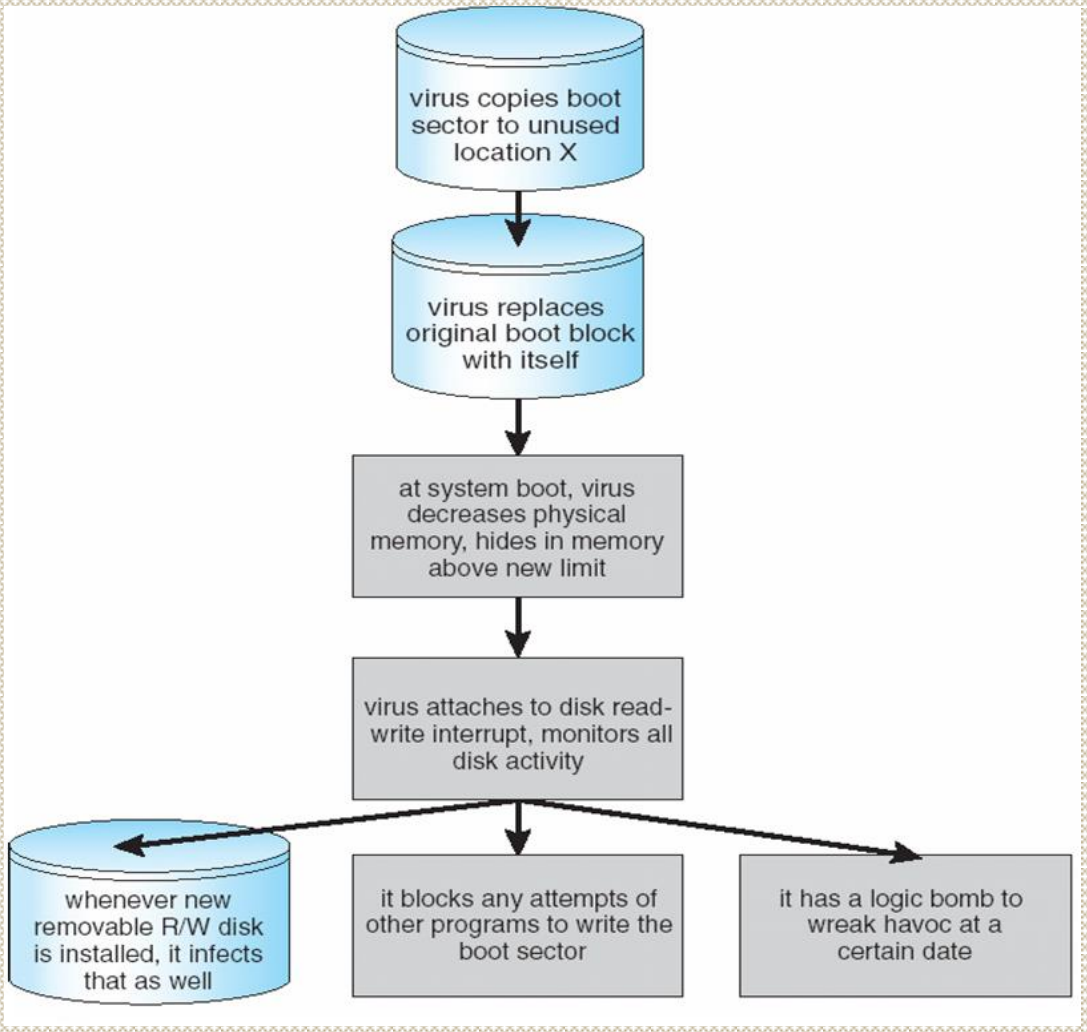
Program Threats (Cont.)

- **Virus dropper** usually a trojan horse inserts virus onto the system.
- Categories of viruses are – a virus can belong to more than one category

- **File** / **parasitic** – infects by appending itself to a file – changes the start of the program to execute its code - after execution returns control to program – its execution is not noticed

- **Boot** / **memory** – infects the boot sector – execute every time system is booted – before OS is loaded – known as memory virus – watches other bootable media and infects them.

A Boot-sector Computer Virus



Program Threats (Cont.)

- **Macro** – written in high level language – visual basic – triggered when a program capable of executing the macro is run - spreadsheets
- **Source code** – looks for source code – modifies it to include virus and help spread virus
- **Polymorphic** – Changes each time it is installed – change does not affect functionality – to avoid having a **virus signature –** a pattern that can be used to identify a virus – series of bytes that make up the virus code
- **Encrypted** – includes decryption code along with encrypted virus to avoid detection

Program Threats (Cont.)

- **Stealth** – avoids detection by modifying parts of the system used to detect – modify read system call so that if the file it has modified is read , the original form of code is returned rather than the infected code

- **Tunneling** – bypass the antivirus scanner by installing in the interrupt handler chain or device drivers

- **Multipartite** – able to infect multiple parts of the system – boot sector, memory

- **Armored** – make it hard for antivirus researchers to understand – compressed to avoid detection

Virus

- Attacks still common, still occurring
- Attacks moved over time from science experiments to tools of organized crime
  - Targeting specific companies
  - Creating botnets to use as tool for spam and DDOS delivery
  - **Keystroke logger** to grab passwords, credit card numbers
- Why is Windows the target for most attacks?
  - Most common
  - Everyone is an administrator
    - Licensing required?
  - Monoculture considered harmful

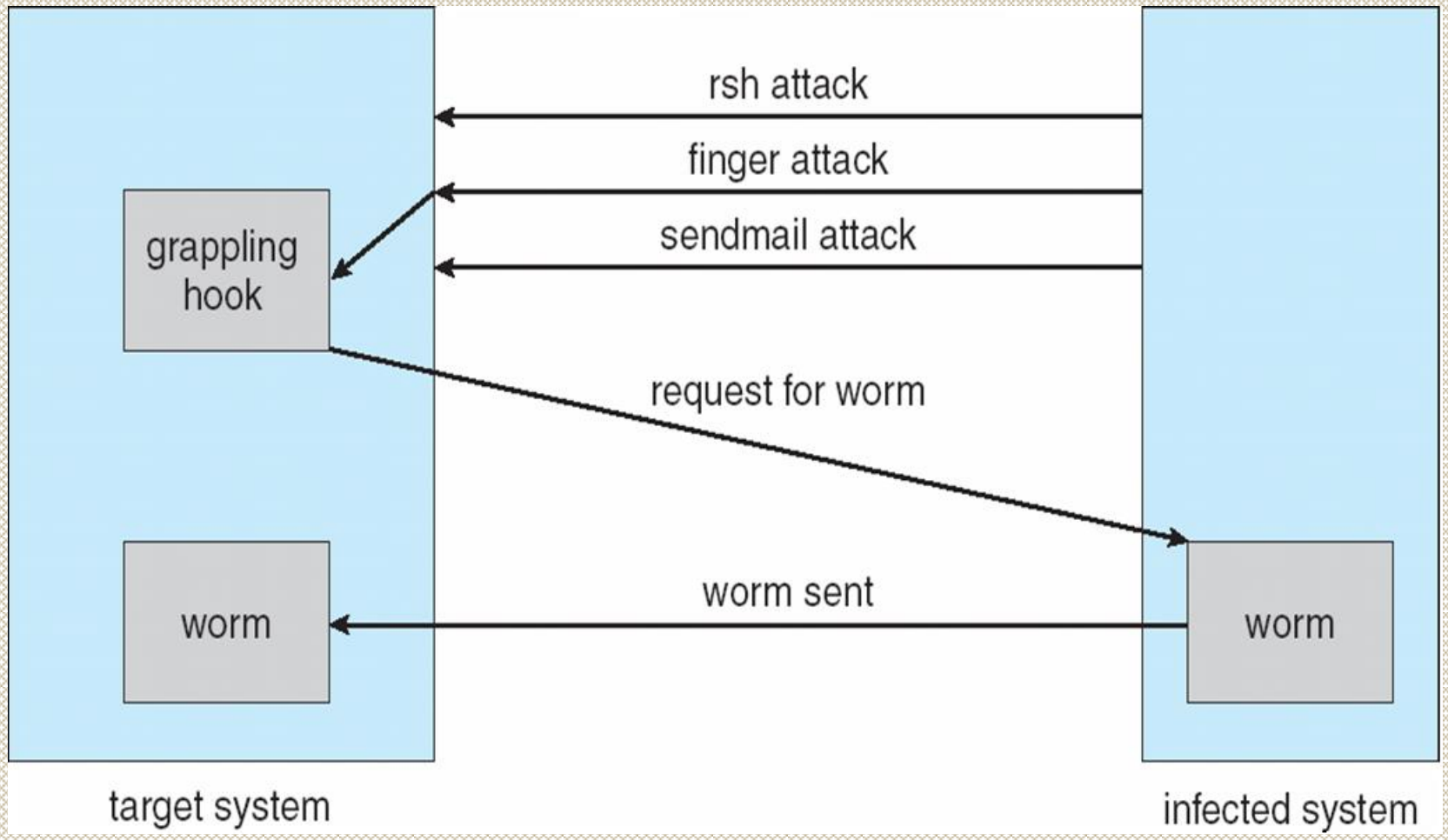
System and Network Threats

- The more "open" a system is – the more services it has enabled - more likely a bug is available.
- OS strive to be secure by default
  - Reduce attack surface – by disabling service by default
- Network threats harder to detect, prevent
  - Protection systems weaker
  - No physical limits once system attached to internet
    - Or on network with system attached to internet
  - Even determining location of connecting system difficult
    - IP address is only knowledge

System and Network Threats (Cont.)

- **Worms** – use **spawn** mechanism; standalone program
- Morris Internet worm
- Exploited UNIX networking features (remote access) and bugs in *finger* and *sendmail* programs – buffer overflow problem.
- Exploited trust-relationship mechanism used by *rsh* to access friendly systems without use of password
- **Grappling hook** program uploaded main worm program
  - 99 lines of C code
- Hooked system then uploaded main code, tried to attack connected systems
- Also tried to break into other users accounts on local system via password guessing
- If target system already infected, abort, except for every 7^{th} time

The Morris Internet Worm



Sobig.F Worm

- More modern example
- Disguised as a photo uploaded to adult newsgroup via account created with stolen credit card
- Targeted Windows systems
- Had own SMTP engine to mail itself as attachment to everyone in infect system's address book
- Disguised with innocuous subject lines, looking like it came from someone known
- Attachment was executable program that created WINPPR23.EXE in default
- It modified windows registry

System and Network Threats (Cont.)

- **Port scanning –** not an attack but means for a cracker to detect the system's vulnerabilities to attack.
- Automated attempt to connect to a range of ports on one or a range of IP addresses
- Detection of answering service protocol
- Detection of OS and version running on system
- `nmap` scans all ports in a given IP range for a response
- `nessus` has a database of protocols and bugs (and exploits) to apply against a system
- Frequently launched from **zombie systems**
  - To decrease trace-ability

System and Network Threats (Cont.)

- **Denial of Service** – not at gaining information or stealing resources but rather at disruption legitimate use of system
- Overload the targeted computer preventing it from doing any useful work
- **Distributed denial-of-service** (**DDOS**) come from multiple sites at once
- Consider the start of the IP-connection handshake (SYN) – I want to start a TCP connection is never followed with the connection is now complete
  - How many started-connections can the OS handle?
- Consider traffic to a web site
  - How can you tell the difference between being a target and being really popular?
- Accidental – CS students writing bad `fork()` code

User Authentication

- Crucial to identify user correctly, as protection systems depend on user ID
- User identity most often established through *passwords*, can be considered a special case of either keys or capabilities
- Passwords must be kept secret
- Frequent change of passwords
- History to avoid repeats
- Use of “non-guessable” passwords
- Log all invalid access attempts (but not the passwords themselves)
- Unauthorized transfer

Passwords

- Passwords may also either be encrypted or allowed to be used only once
  - Does encrypting passwords solve the exposure problem?
    - Might solve **sniffing**
    - Consider **shoulder surfing**
    - Consider Trojan horse keystroke logger
    - How are passwords stored at authenticating site?

User Authentication

- **Encrypt to avoid having to keep secret**
  - But keep secret anyway (i.e. Unix uses superuser-only readably file `/etc/shadow`)
  - Use algorithm easy to compute but difficult to invert
  - Only encrypted password stored, never decrypted
  - Add "salt" to avoid the same password being encrypted to the same value
- **One-time passwords**
  - Use a function based on a seed to compute a password, both user and computer
  - Hardware device / calculator / key fob to generate the password
    - Changes very frequently

User Authentication

- **Biometrics**
  - Some physical attribute (fingerprint, hand scan)
  - Finger print readers are more accurate and cost effective
- **Multi-factor authentication**
  - Need two or more factors for authentication
    - i.e. USB "dongle", biometric measure, and password