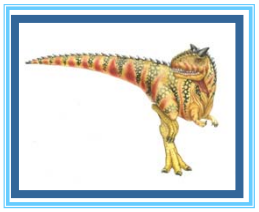
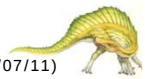
Chapter 15: Security



(slides improved by R. Doemer, 03/07/11)

Chapter 15: Security

- The Security Problem
- Program Threats
- System and Network Threats
- Cryptography as a Security Tool
- User Authentication
- Implementing Security Defenses
- Firewalling to Protect Systems and Networks
- Computer-Security Classifications
- An Example: Windows XP



(slide modified by R. Doemer, 03/07/11)

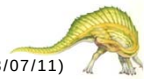
The Security Problem

- **Security**
  - must consider **external environment** of the system
  - **protect** the system resources
- Terminology
  - **Intruders** (crackers) attempt to breach security
  - **Threat** is potential security violation
  - **Attack** is attempt to breach security
    - Attack can be accidental or malicious
    - It's easier to protect against accidental than malicious misuse

(slide modified by R. Doemer, 03/07/11)

Security Violations

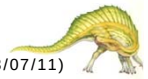
- Categories
  - **Breach of confidentiality**
    - Unauthorized reading of data
  - **Breach of integrity**
    - Unauthorized modification of data
  - **Breach of availability**
    - Unauthorized destruction of data
  - **Theft of service**
    - Unauthorized use of resources
  - **Denial of service**
    - Preventing legitimate use of a service
    - E.g. distributed denial-of-service (DDOS) attacks



(slide improved by R. Doemer, 03/07/11)

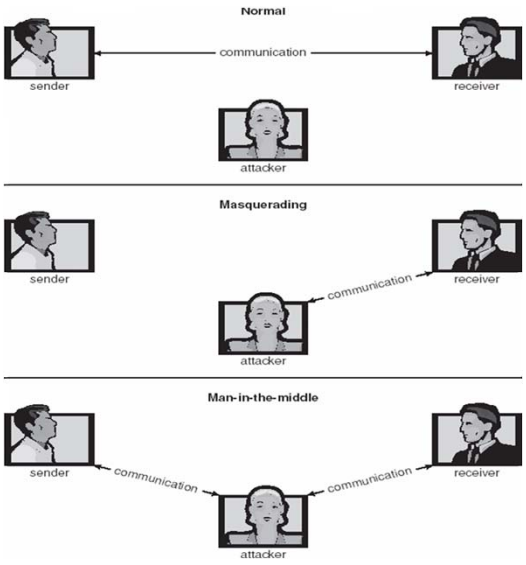
Security Violations

- Methods
  - **Masquerading** (to breach **authentication**)
    - Participant in a communication pretends to be someone else (see next slide)
  - **Replay attack**
    - Malicious or fraudulent repeat of a valid transmission
    - Message modification
  - **Man-in-the-middle attack**
    - Attacker between sender and receiver, masquerades as sender to the receiver, and vice versa (see next slide)
  - **Session hijacking**
    - Intercepting an active communication



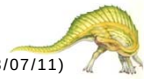
(slide improved by R. Doemer, 03/07/11)

Standard Security Attacks



Security Measures at Four Levels

- **Security** must occur at four **levels** to be effective:
  - **Physical**
    - Lock access to machine rooms, terminals, etc.
  - **Human**
    - Assure that only appropriate users have access
    - Avoid **social engineering, phishing, dumpster diving**
  - **Operating System**
    - Protection mechanisms!
  - **Network**
    - Secure communication (more important than ever!)
- Security is as weak as the weakest line in the chain!



(slide improved by R. Doemer, 03/07/11)

Common Program Threats

- **Trojan Horse**
  - Code segment that misuses its environment
  - Exploits mechanisms for allowing programs written by users to be executed by other users
    - Be careful about your search path in your shell ($PATH)!
  - **Spyware**, pop-up browser windows, covert channels
- **Trap Door**
  - Specific user identifier or password that circumvents normal security procedures
  - Could be included in a compiler
- **Logic Bomb**
  - Program that initiates a security incident under certain circumstances
    - E.g. specific time, or when no longer in employee list
- **Stack and Buffer Overflow**
  - Exploits a bug in a program (overflow either the stack or memory buffers) (see next slides)

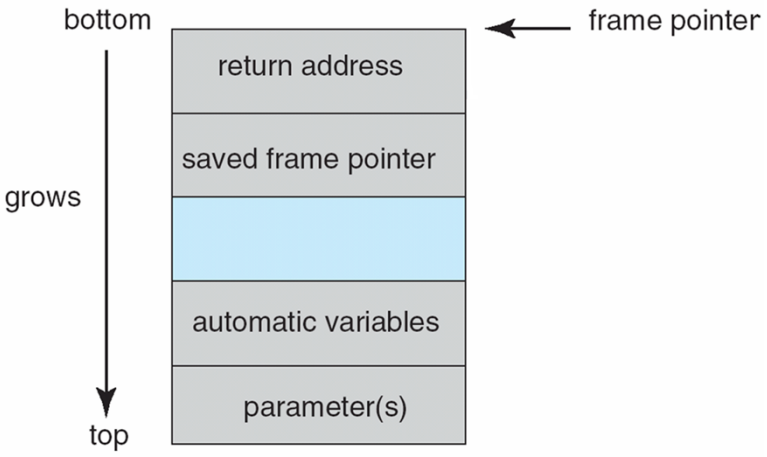
(slide modified by R. Doemer, 03/07/11)

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]);
        /* ... do something */
        return 0;
   }
}
```

(slide modified by R. Doemer, 03/07/11)

Layout of a Typical Stack Frame

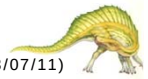


(slide modified by R. Doemer, 03/07/11)

Short Code Segment to start a Shell

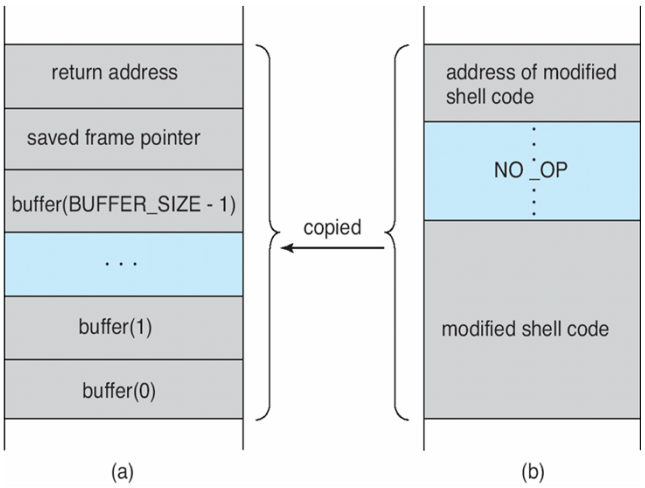
```
#include <stdio.h>
int main(int argc, char *argv[])
{
   execl("\bin\sh", "\bin\sh", NULL);
   return 0;
}
```

- **Attacker**
  - Writes the above program
    - which starts an interactive shell with all permissions of its environment
  - Compiles the program into assembly code
  - Manipulates binary to fit the target stack frame, see next slide!
  - Feeds binary as argument to the program with buffer-overflow condition
  - Gains full interactive access!



(slide improved by R. Doemer, 03/07/11)

Hypothetical Stack Frame



Before attack

At time of attack

(slide modified by R. Doemer, 03/07/11)

Computer Viruses

- **Virus**
  - Code fragment embedded in legitimate program
  - Very specific to CPU architecture, operating system, applications
  - For example, borne via email or as a *macro* in documents
    - e.g. 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
```

- **Virus dropper** inserts virus onto the system
  - e.g. a Trojan horse, or
  - infected disk, memory stick

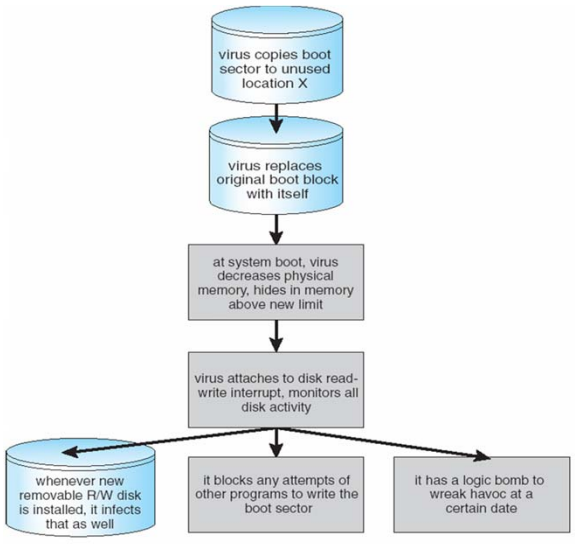
(slide modified by R. Doemer, 03/07/11)

Computer Viruses

- Many **categories of viruses** exist, literally many thousands of viruses
  - File
  - Boot
  - Macro
  - Source code
  - Polymorphic
  - Encrypted
  - Stealth
  - Tunneling
  - Multipartite
  - Armored
  - …

(slide modified by R. Doemer, 03/07/11)

A Boot-Sector Computer Virus



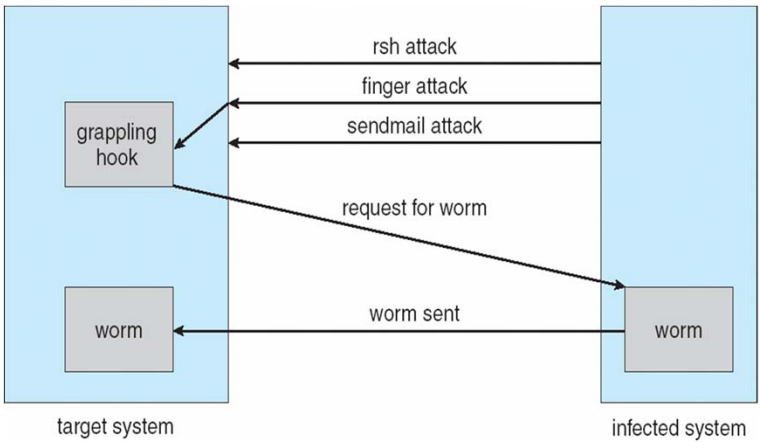
System and Network Threats

- **Worms** – use **spawn** mechanism; standalone program
- Example: **Morris internet worm**
  - Exploited UNIX networking features (remote shell, *rsh*) and bugs in *finger* and *sendmail* programs
  - **Grappling hook** program uploaded main worm program (see next slide)
- **Port scanning**
  - Automated attempt to connect to a range of ports on one or a range of IP addresses
- **Denial of Service**
  - *Overload* the targeted computer preventing it from doing any useful work
  - **Distributed denial-of-service** (**DDOS**) attacks come from multiple sites at once

(slide modified by R. Doemer, 03/07/11)

The Morris Internet Worm

- **1988, Cornell University**, Robert Morris, Jr.
  - Exploited flaws in the Unix operating system



(slide modified by R. Doemer, 03/07/11)