Chapter 15: Security

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

Objectives

- To discuss security threats and attacks
- To explain the fundamentals of encryption, authentication, and hashing
- To examine the uses of cryptography in computing
- To describe the various countermeasures to security attacks

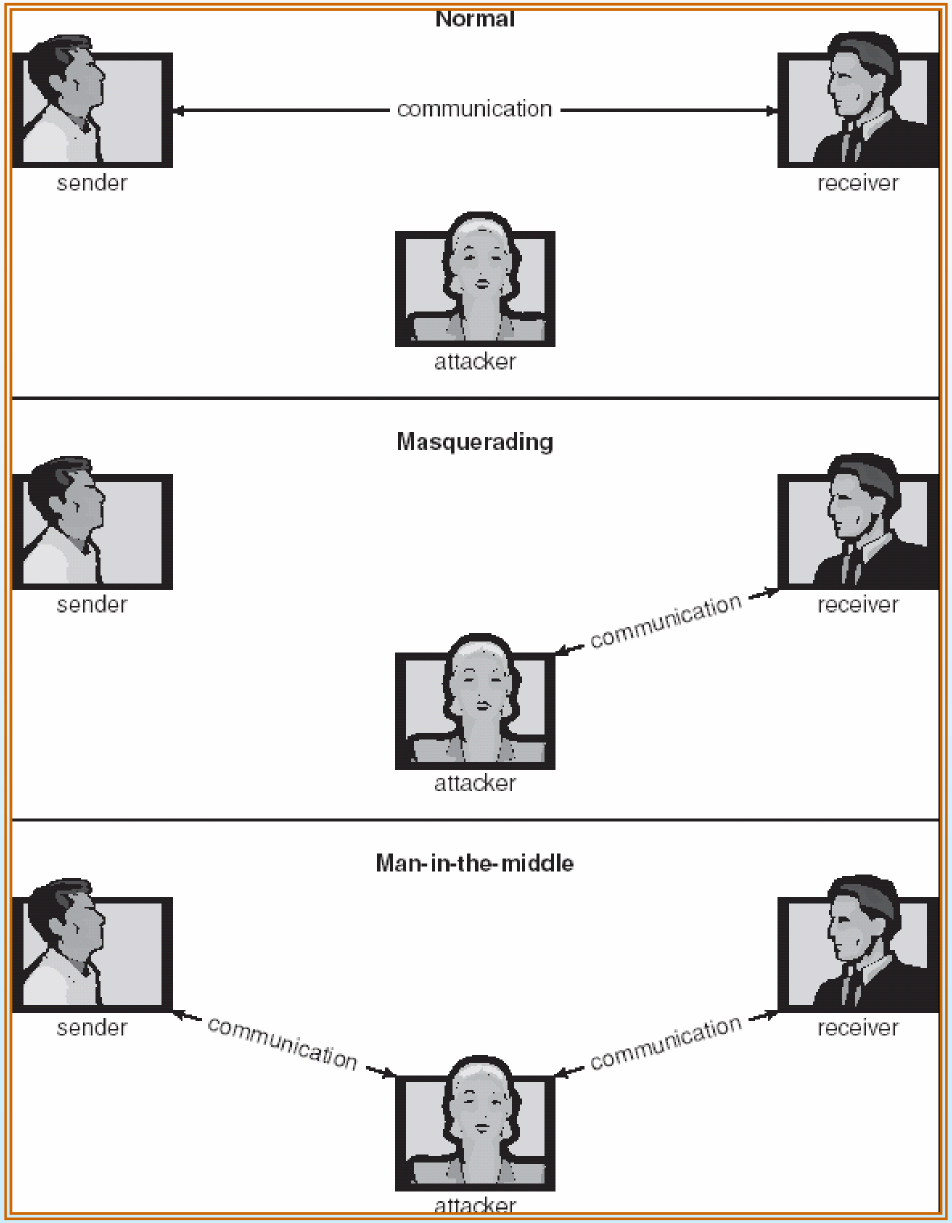
The Security Problem

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

Security Violations

- Categories
  - **Breach of confidentiality**
  - **Breach of integrity**
  - **Breach of availability**
  - **Theft of service**
  - **Denial of service**
- Methods
  - **Masquerading (breach authentication)**
  - **Replay attack**
    - **Message modification**
  - **Man-in-the-middle attack**
  - **Session hijacking**

Standard Security Attacks



Security Measure Levels

- Security must occur at four levels to be effective:
  - Physical
  - Human
    - Avoid **social engineering, phishing, dumpster diving**
  - Operating System
  - Network
- Security is as week as the weakest chain

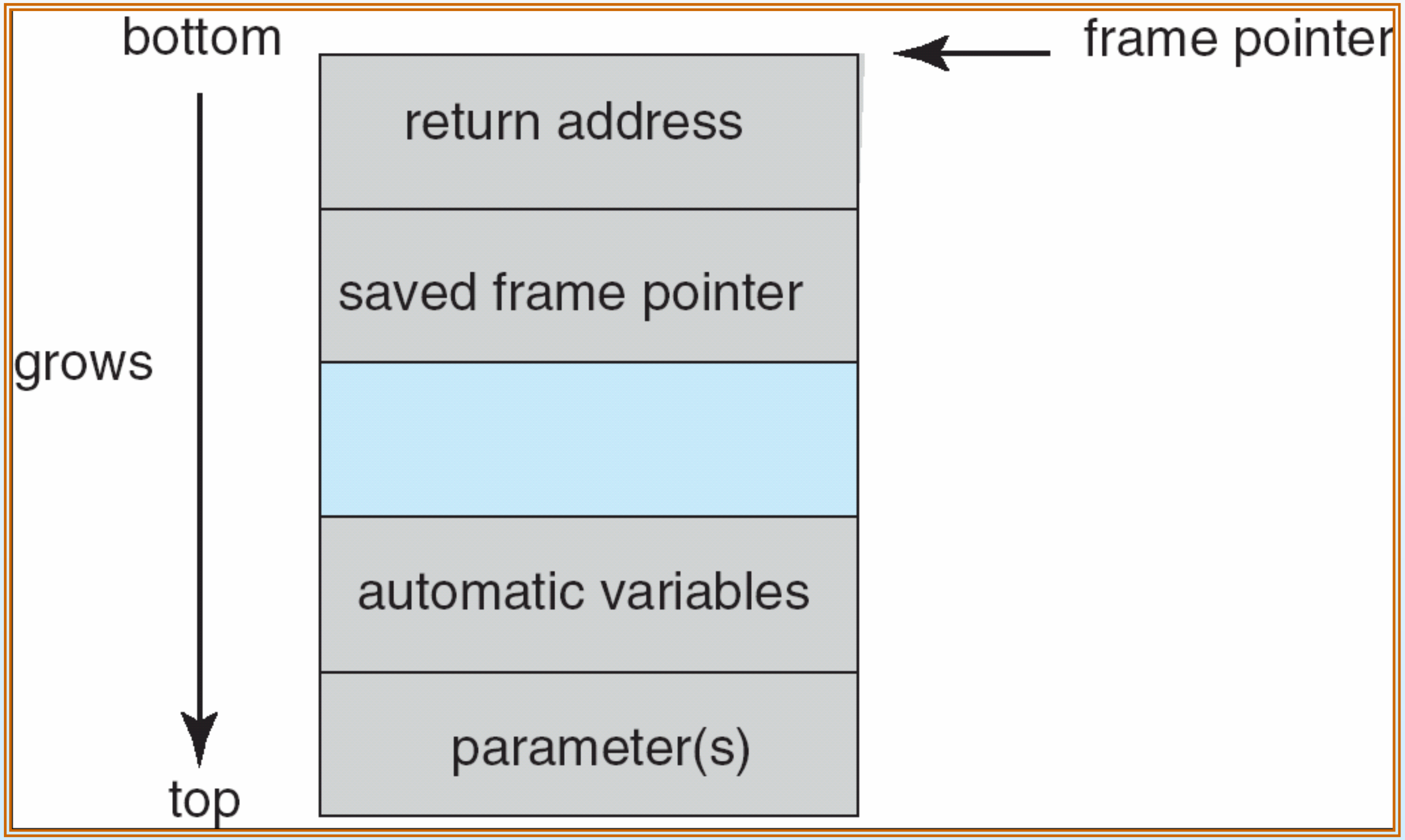
Program Threats

- Trojan Horse
  - Code segment that misuses its environment
  - Exploits mechanisms for allowing programs written by users to be executed by other users
  - **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
- Stack and Buffer Overflow
  - Exploits a bug in a program (overflow either the stack or memory buffers)

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;
   }
}
```

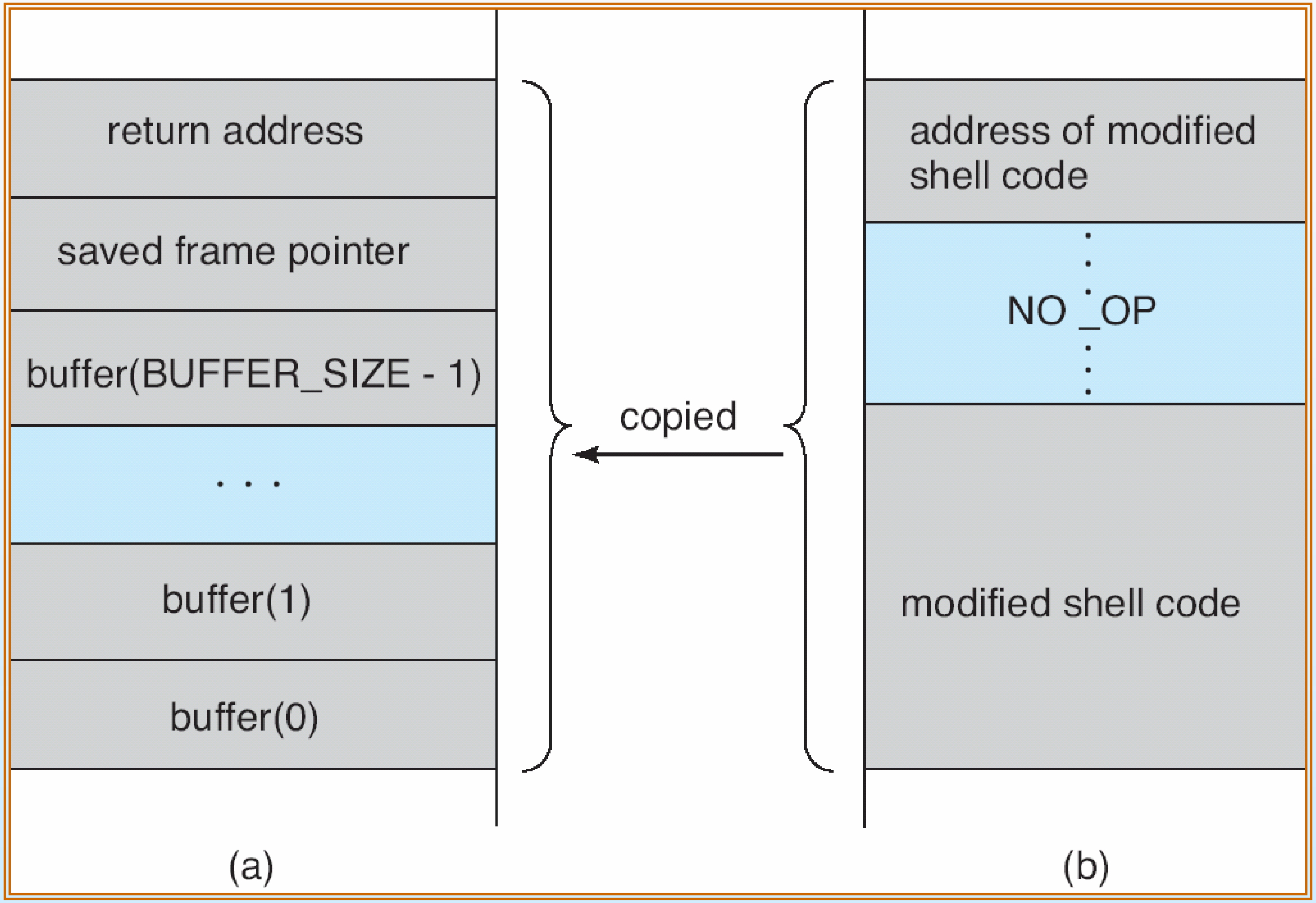
Layout of Typical Stack Frame



Modified Shell Code

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

Hypothetical Stack Frame



Before attack

After attack

Program Threats (Cont.)

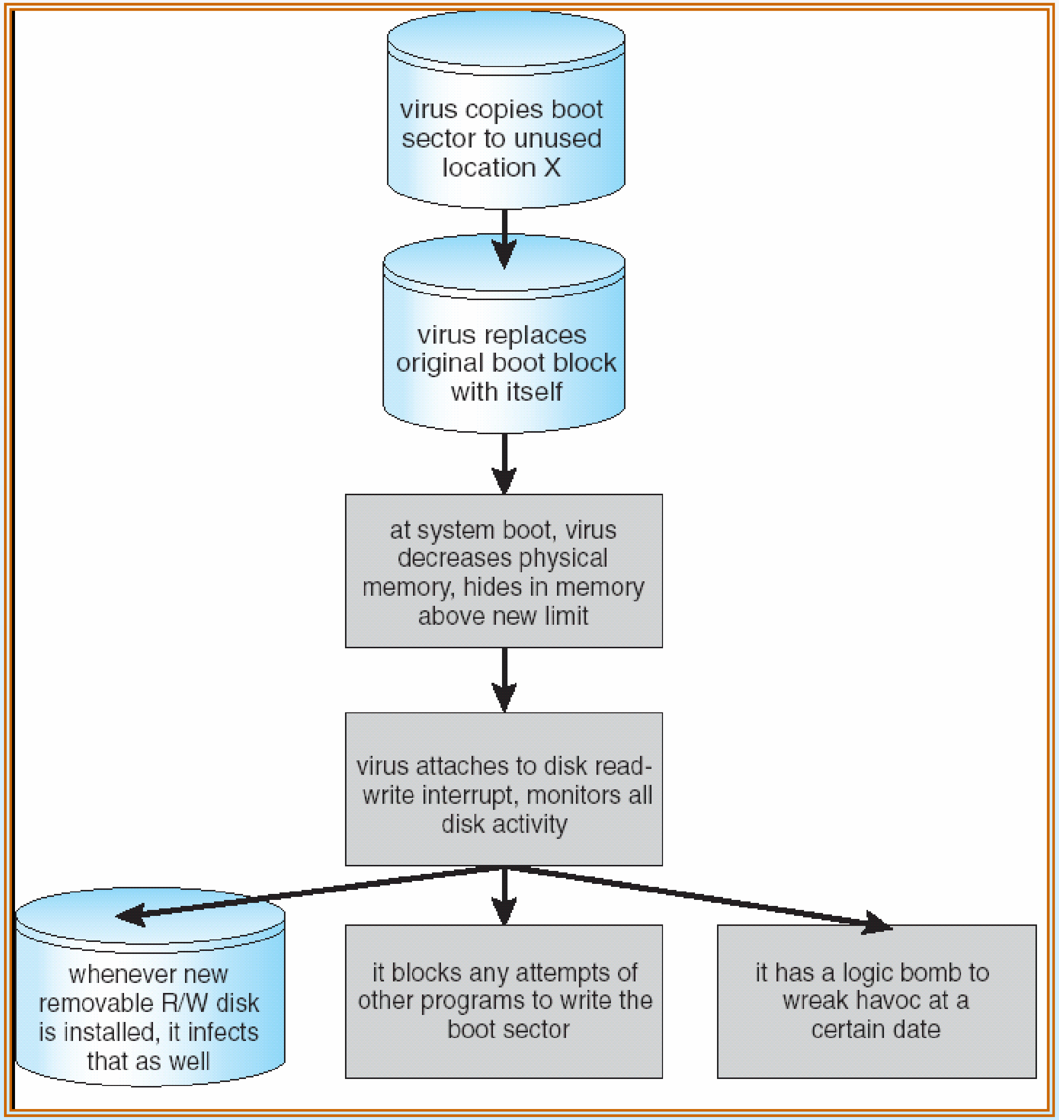
- Viruses
  - Code fragment embedded in legitimate program
  - Very specific to CPU architecture, operating system, applications
  - Usually borne via email or as a macro
    - Visual Basic Macro to reformat hard drive

```
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** inserts virus onto the system
- Many categories of viruses, literally many thousands of viruses
  - File
  - Boot
  - Macro
  - Source code
  - Polymorphic
  - Encrypted
  - Stealth
  - Tunneling
  - Multipartite
  - Armored

A Boot-sector Computer Virus



virus copies boot sector to unused location X

virus replaces original boot block with itself

at system boot, virus decreases physical memory, hides in memory above new limit

virus attaches to disk read-write interrupt, monitors all disk activity

whenever new removable R/W disk is installed, it infects that as well

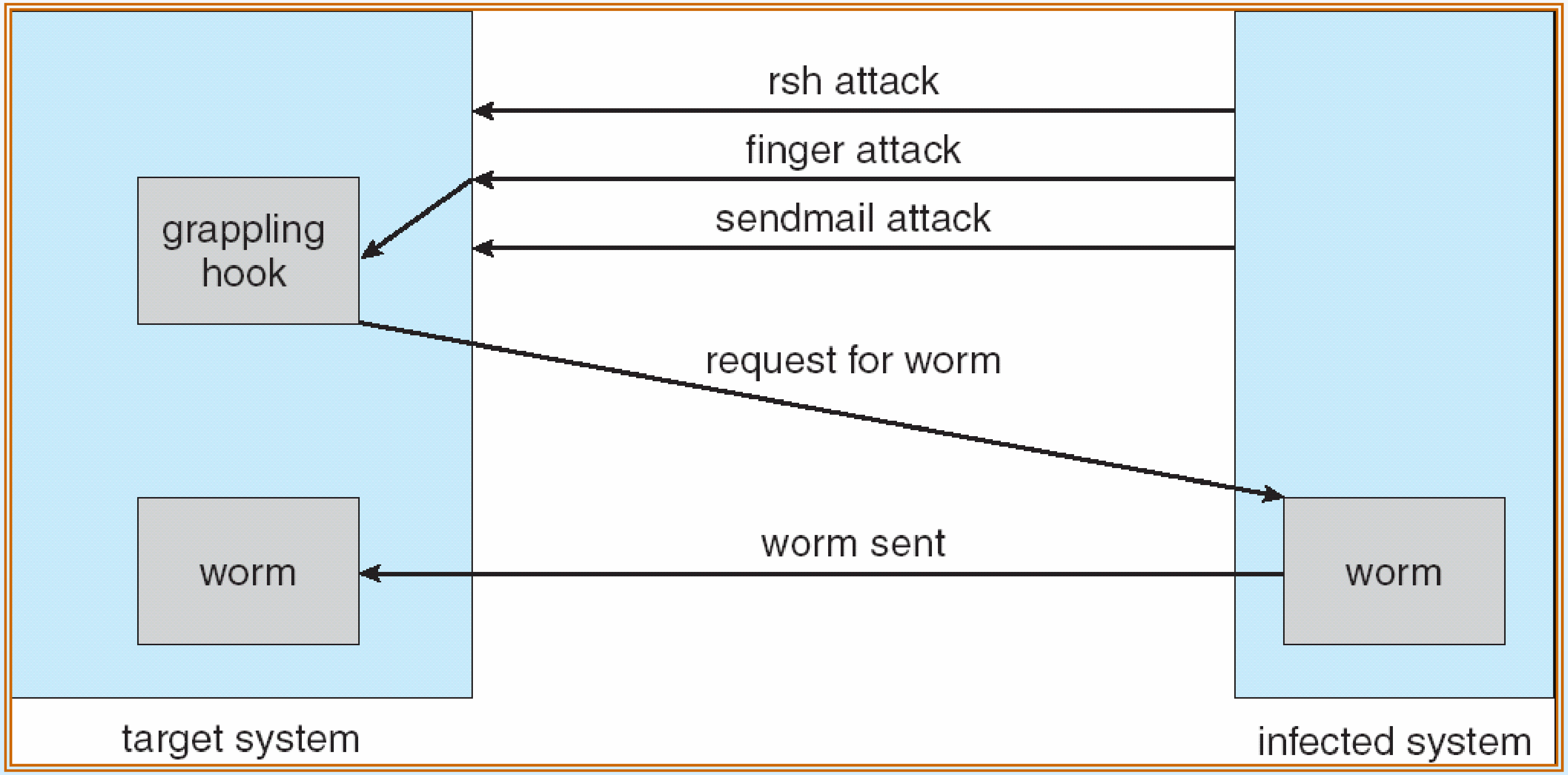
it blocks any attempts of other programs to write the boot sector

it has a logic bomb to wreak havoc at a certain date

System and Network Threats

- Worms – use **spawn** mechanism; standalone program
- Internet worm
  - Exploited UNIX networking features (remote access) and bugs in *finger* and *sendmail* programs
  - **Grappling hook** program uploaded main worm program
- 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**) come from multiple sites at once

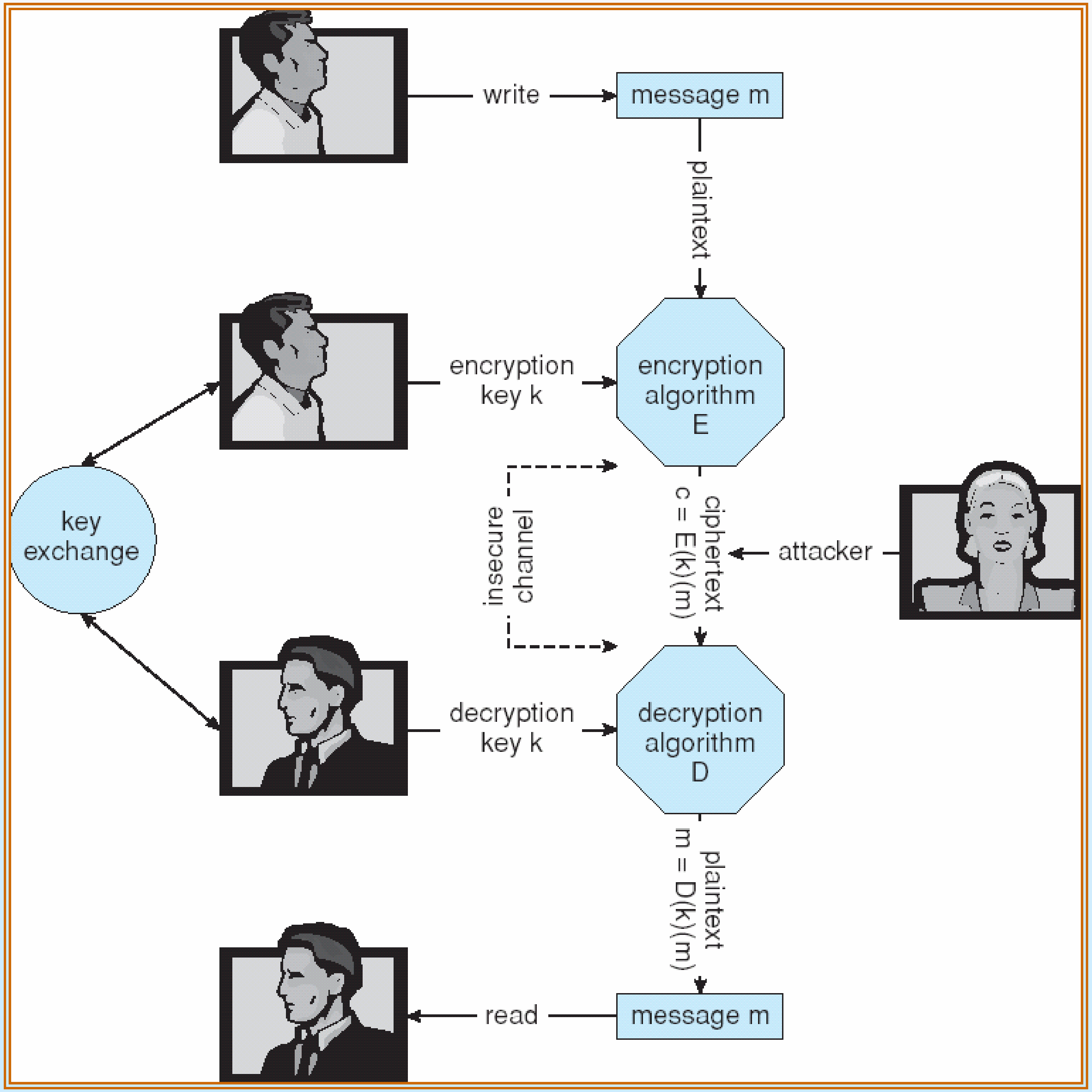
The Morris Internet Worm



Cryptography as a Security Tool

- Broadest security tool available
  - Source and destination of messages cannot be trusted without cryptography
  - Means to constrain potential senders (*sources*) and / or receivers (*destinations*) of *messages*
- Based on secrets (**keys**)

Secure Communication over Insecure Medium



Encryption

- Encryption algorithm consists of
  - Set of K keys
  - Set of M Messages
  - Set of C ciphertexts (encrypted messages)
  - A function $E : K \rightarrow (M \rightarrow C)$. That is, for each $k \in K$, $E(k)$ is a function for generating ciphertexts from messages.
    - Both E and $E(k)$ for any k should be efficiently computable functions.
  - A function $D : K \rightarrow (C \rightarrow M)$. That is, for each $k \in K$, $D(k)$ is a function for generating messages from ciphertexts.
    - Both D and $D(k)$ for any k should be efficiently computable functions.
- An encryption algorithm must provide this essential property: Given a ciphertext $c \in C$, a computer can compute m such that $E(k)(m) = c$ only if it possesses $D(k)$.
  - Thus, a computer holding $D(k)$ can decrypt ciphertexts to the plaintexts used to produce them, but a computer not holding $D(k)$ cannot decrypt ciphertexts.
  - Since ciphertexts are generally exposed (for example, sent on the network), it is important that it be infeasible to derive $D(k)$ from the ciphertexts

Symmetric Encryption

- Same key used to encrypt and decrypt
  - *E*(*k*) can be derived from *D*(*k*), and vice versa
- DES is most commonly used symmetric block-encryption algorithm (created by US Govt)
  - Encrypts a block of data at a time
- Triple-DES considered more secure
- Advanced Encryption Standard (**AES**), **twofish** up and coming
- RC4 is most common symmetric stream cipher, but known to have vulnerabilities
  - Encrypts/decrypts a stream of bytes (i.e wireless transmission)
  - Key is a input to psuedo-random-bit generator
    - Generates an infinite **keystream**

Asymmetric Encryption

- Public-key encryption based on each user having two keys:
  - public key – published key used to encrypt data
  - private key – key known only to individual user used to decrypt data
- Must be an encryption scheme that can be made public without making it easy to figure out the decryption scheme
  - Most common is RSA block cipher
  - Efficient algorithm for testing whether or not a number is prime
  - No efficient algorithm is know for finding the prime factors of a number

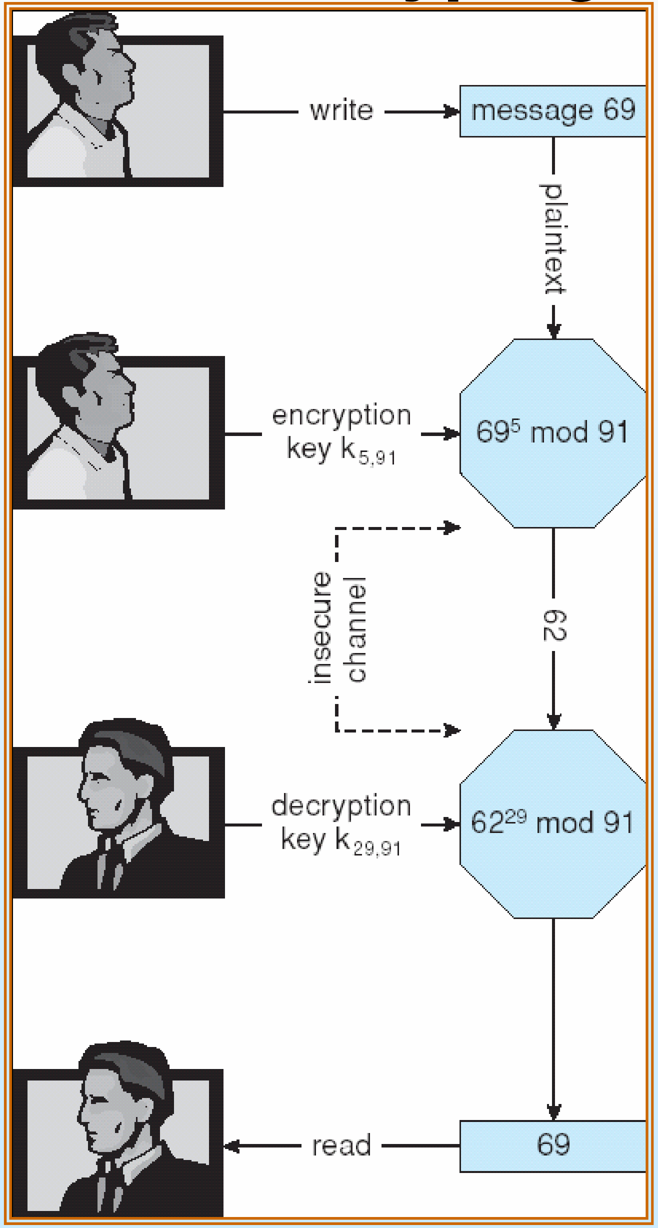
Asymmetric Encryption (Cont.)

- Formally, it is computationally infeasible to derive $D(k_d , N)$ from $E(k_e , N)$, and so $E(k_e , N)$ need not be kept secret and can be widely disseminated
  - $E(k_e , N)$ (or just k_e) is the **public key**
  - $D(k_d , N)$ (or just k_d) is the **private key**
  - N is the product of two large, randomly chosen prime numbers p and q (for example, p and q are 512 bits each)
  - Encryption algorithm is $E(k_e , N)(m) = m^{k_e} \bmod N$, where k_e satisfies $k_e k_d \bmod (p-1)(q-1) = 1$
  - The decryption algorithm is then $D(k_d , N)(c) = c^{k_d} \bmod N$

Asymmetric Encryption Example

- For example. make p = 7and q = 13
- We then calculate $N = 7 \ast 13 = 91$ and $(p-1)(q-1) = 72$
- We next select k_e relatively prime to 72 and< 72, yielding 5
- Finally,we calculate k_d such that $k_e k_d$ mod 72 = 1, yielding 29
- We how have our keys
  - Public key, $k_{e,}$ N = 5, 91
  - Private key, k_d , N = 29, 91
- Encrypting the message 69 with the public key results in the cyphertext 62
- Cyphertext can be decoded with the private key
  - Public key can be distributed in cleartext to anyone who wants to communicate with holder of public key

Encryption and Decryption using RSA Asymmetric Cryptography



Cryptography (Cont.)

- Note symmetric cryptography based on transformations, asymmetric based on mathematical functions
  - Asymmetric much more compute intensive
  - Typically not used for bulk data encryption

Authentication

- Constraining set of potential senders of a message
  - Complementary and sometimes redundant to encryption
  - Also can prove message unmodified
- Algorithm components
  - A set K of keys
  - A set M of messages
  - A set A of authenticators
  - A function $S : K \rightarrow (M \rightarrow A)$
    - That is, for each $k \in K$, $S(k)$ is a function for generating authenticators from messages
    - Both S and $S(k)$ for any k should be efficiently computable functions
  - A function $V : K \rightarrow (M \times A \rightarrow \{\text{true, false}\})$. That is, for each $k \in K$, $V(k)$ is a function for verifying authenticators on messages
    - Both V and $V(k)$ for any k should be efficiently computable functions

Authentication (Cont.)

- For a message m, a computer can generate an authenticator $a \in A$ such that $V(k)(m, a)$ = `true` only if it possesses $S(k)$
- Thus, computer holding $S(k)$ can generate authenticators on messages so that any other computer possessing $V(k)$ can verify them
- Computer not holding $S(k)$ cannot generate authenticators on messages that can be verified using $V(k)$
- Since authenticators are generally exposed (for example, they are sent on the network with the messages themselves), it must not be feasible to derive $S(k)$ from the authenticators

Authentication – Hash Functions

- Basis of authentication
- Creates small, fixed-size block of data (**message digest, hash value**) from *m*
- Hash Function *H* must be collision resistant on *m*
  - Must be infeasible to find an *m'* ≠ *m* such that *H*(*m*) = *H*(*m'*)
- If *H*(*m*) = *H*(*m'*), then *m* = *m*'
  - The message has not been modified
- Common message-digest functions include **MD5**, which produces a 128-bit hash, and **SHA-1**, which outputs a 160-bit hash

Authentication - MAC

- Symmetric encryption used in **message-authentication code** (**MAC**) authentication algorithm
- Simple example:
  - MAC defines $S(k)(m) = f\,(k, H(m))$
    - Where f is a function that is one-way on its first argument
      - k cannot be derived from $f\,(k, H(m))$
    - Because of the collision resistance in the hash function, reasonably assured no other message could create the same MAC
    - A suitable verification algorithm is $V(k)(m, a) \equiv (\,f\,(k,m) = a)$
    - Note that k is needed to compute both $S(k)$ and $V(k)$, so anyone able to compute one can compute the other

Authentication – Digital Signature

- Based on asymmetric keys and digital signature algorithm
- Authenticators produced are **digital signatures**
- In a digital-signature algorithm, computationally infeasible to derive $S(k_s)$ from $V(k_v)$
  - V is a one-way function
  - Thus, k_v is the public key and k_s is the private key
- Consider the RSA digital-signature algorithm
  - Similar to the RSA encryption algorithm, but the key use is reversed
  - Digital signature of message $S(k_s)(m) = H(m)^{k_s} \bmod N$
  - The key k_s again is a pair d, N, where N is the product of two large, randomly chosen prime numbers p and q
  - Verification algorithm is $V(k_v)(m, a) \equiv (a^{k_v} \bmod N = H(m))$
    - Where k_v satisfies $k_v k_s \bmod (p - 1)(q - 1) = 1$

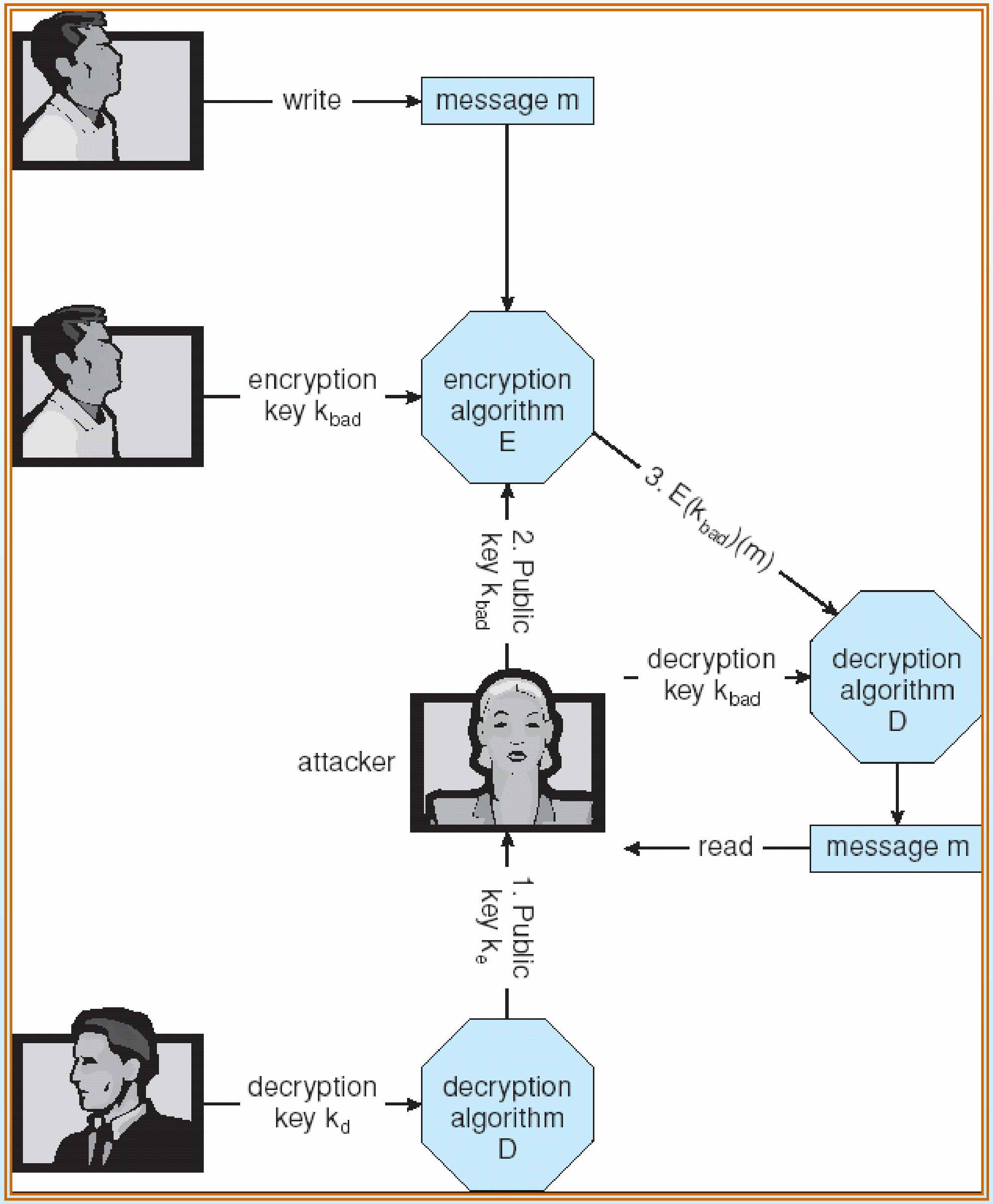
Authentication (Cont.)

- Why authentication if a subset of encryption?
  - Fewer computations (except for RSA digital signatures)
  - Authenticator usually shorter than message
  - Sometimes want authentication but not confidentiality
    - Signed patches et al
  - Can be basis for **non-repudiation**

Key Distribution

- Delivery of symmetric key is huge challenge
  - Sometimes done **out-of-band**
- Asymmetric keys can proliferate – stored on **key ring**
  - Even asymmetric key distribution needs care – man-in-the-middle attack

Man-in-the-middle Attack on Asymmetric Cryptography



Digital Certificates

- Proof of who or what owns a public key
- Public key digitally signed a trusted party
- Trusted party receives proof of identification from entity and certifies that public key belongs to entity
- Certificate authority are trusted party – their public keys included with web browser distributions
  - They vouch for other authorities via digitally signing their keys, and so on

Encryption Example - SSL

- Insertion of cryptography at one layer of the ISO network model (the transport layer)
- SSL – Secure Socket Layer (also called TLS)
- Cryptographic protocol that limits two computers to only exchange messages with each other
  - Very complicated, with many variations
- Used between web servers and browsers for secure communication (credit card numbers)
- The server is verified with a **certificate** assuring client is talking to correct server
- Asymmetric cryptography used to establish a secure **session key** (symmetric encryption) for bulk of communication during session
- Communication between each computer theb uses symmetric key cryptography

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
  - Also can include something user has and /or a user attribute
- Passwords must be kept secret
  - Frequent change of passwords
  - Use of “non-guessable” passwords
  - Log all invalid access attempts
- Passwords may also either be encrypted or allowed to be used only once

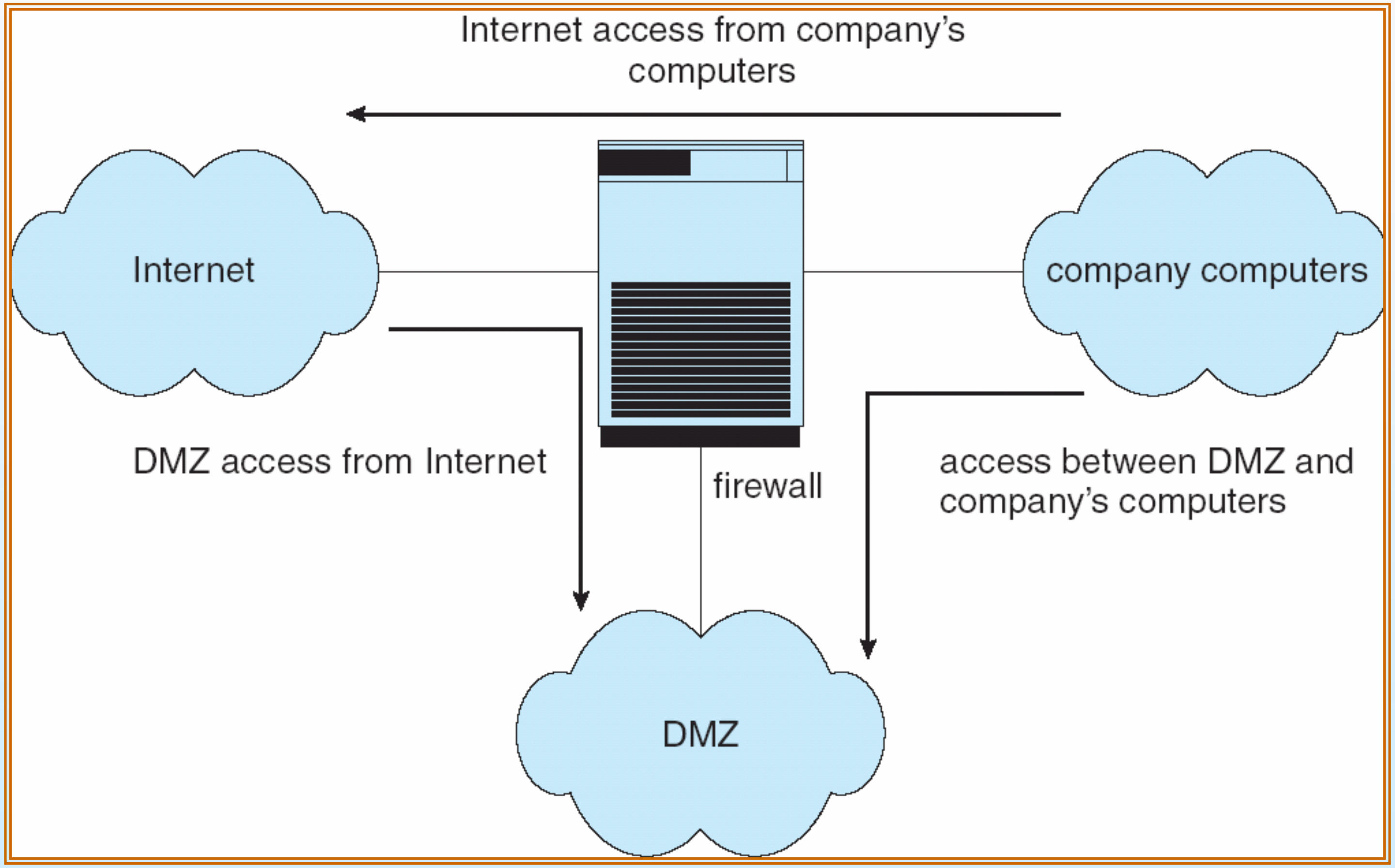
Implementing Security Defenses

- **Defense in depth** is most common security theory – multiple layers of security
- Security policy describes what is being secured
- Vulnerability assessment compares real state of system / network compared to security policy
- Intrusion detection endeavors to detect attempted or successful intrusions
  - **Signature-based** detection spots known bad patterns
  - **Anomaly detection** spots differences from normal behavior
    - Can detect **zero-day** attacks
  - **False-positives** and **false-negatives** a problem
- Virus protection
- Auditing, accounting, and logging of all or specific system or network activities

Firewalling to Protect Systems and Networks

- A network firewall is placed between trusted and untrusted hosts
  - The firewall limits network access between these two security domains
- Can be tunneled or spoofed
  - Tunneling allows disallowed protocol to travel within allowed protocol (i.e. telnet inside of HTTP)
  - Firewall rules typically based on host name or IP address which can be spoofed
- **Personal firewall** is software layer on given host
  - Can monitor / limit traffic to and from the host
- **Application proxy firewall** understands application protocol and can control them (i.e. SMTP)
- **System-call firewall** monitors all important system calls and apply rules to them (i.e. this program can execute that system call)

Network Security Through Domain Separation Via Firewall


Internet access from company's computers
Internet
company computers
DMZ access from Internet
firewall
access between DMZ and company's computers
DMZ

Computer Security Classifications

- U.S. Department of Defense outlines four divisions of computer security: **A**, **B**, **C**, and **D**.
- **D** – Minimal security.
- **C** – Provides discretionary protection through auditing. Divided into **C1** and **C2**. **C1** identifies cooperating users with the same level of protection. **C2** allows user-level access control.
- **B** – All the properties of **C**, however each object may have unique sensitivity labels. Divided into **B1**, **B2**, and **B3**.
- **A** – Uses formal design and verification techniques to ensure security.

Example: Windows XP

- Security is based on user accounts
  - Each user has unique security ID
  - Login to ID creates **security access token**
    - Includes security ID for user, for user’s groups, and special privileges
    - Every process gets copy of token
    - System checks token to determine if access allowed or denied
- Uses a subject model to ensure access security. A subject tracks and manages permissions for each program that a user runs
- Each object in Windows XP has a security attribute defined by a security descriptor
  - For example, a file has a security descriptor that indicates the access permissions for all users

End of Chapter 15