Security (Chapter 15)
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: accidental or malicious
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

Normal

Sender → Communication → Receiver

Attacker

Masquerading

Sender → Communication → Receiver

Attacker

Man-in-the-middle

Sender → Communication → Receiver

Attacker

Sender → Communication → Attacker → Communication → Receiver
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 weak as the weakest link in the 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)
#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

- bottom
- return address
- saved frame pointer
- automatic variables
- parameter(s)
- top
- frame pointer

The stack frame growsbottom and has a frame pointer that points to the top of the stack.
#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
      ```vbc
      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

- **gripping hook**
- **worm**

**target system**

**infected system**

- rsh attack
- finger attack
- sendmail attack
- request for worm
- worm sent
Implementing Security Defenses

- Defense in depth is most common security theory – multiple layers of security
- Solutions ranging from education, through technology to writing bug-free code

Components needed:
- 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 (previously unknown methods of intrusion)
  - 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
Network Security Through Domain Separation Via Firewall

Semi-trusted and semi-secure domain
Security Issues on the Internet: Botnets and Other Malware
Outline

- What are botnets
- Working of botnets
- Measuring Botnets
- ...

15.20 Silberschatz, Galvin and Gagne ©2009
Botnets

- A network of compromised computers that have been infected with malicious code, and can be remotely-controlled through commands sent via the Internet by a botmaster
- Used for making money! Who pays:
  - Internet Advertising companies for downloading adware onto vulnerable PCs
  - Companies who send spam, viruses and other malware
  - Google (unintentionally, see Clickbot. A summary)
- Used for the heck of it
Botnets Work

- Distributed DoS: Estonia Spring 2007
  - Government web sites: 1000 visits/day -> 2000 visits/second
  - Attack lasted weeks (instead of hours), high rate packet attacks
  - NATO and US sent computer security experts
- Spamming: 80% of spam due to botnets
- Click fraud attacks
- Identity thefts:
  - MasterCard reported in 2005 > 40 million credit card numbers accessed via Internet by hackers
- Cheating in online polls/games
- … many more
Botnet Recruiting (1)

- Email
  - Requires user interaction, social engineering
  - Easiest method; common.

- Instant message
  - Internet Relay Chat (IRC): real-time Internet chat or synchronous conferencing, designed for group communication
    - Name comes from IRC bots: automatic IRC clients
  - Various: social eng., file transfer, vulnerabilities

- Remote software vulnerability
  - Often, no interaction needed
Botnet Recruiting (2)

- Malicious code hosted on websites
  - During the first half of 2006, the Microsoft Security Team reported that it had removed 10 million pieces of malicious software from nearly 4 million computers and web servers.
  - Google tested several million web pages for the presence of malicious software, and determined that 4.5 million of the web pages examined were suspicious in nature. After further testing of the 4.5 million web pages, over 1 million were found to launch downloads of malicious software, and more than two thirds of those programs were “bot” software that, among other things, collected data on banking transactions and then emailed the information to a temporary email account.
  - Due to:
    - Web server security problems
    - User contributed content
    - Advertising and third-party widgets
Botnet Recruiting

Terminology:
- Botmaster
- Command & control servers
- Zombie

(*) : Optional Step
Rallying

- A first task of zombies is rallying
  - how can victims contact the master *safely*?
- Simple, naïve approach:
  - Victims contact single IP, website, ping a server, etc.
  - Easily defeated (ISP intervention, blackhole routing, etc.)
  - Still used by kiddies, first-time malware authors
- Resilient Networks needed
  - IRC: resists assaults by ISPs; very little supervision/intrusion
  - P2P
- Newer botnets use command and control hierarchy, with botmaster, lieutenants, and individual zombies
Challenges

- Determining the source of a botnet-based attack is challenging:
  - Every zombie host is an attacker
  - Botnets can exist in a benign state for an arbitrary amount of time before they are used for a specific attack
  - Traditional approach: identify the C&C server and disable it
  - New trend: P2P networks, C&C server anonymized among the other peers (zombies)
- Measuring the size of botnets
The Shadowserver Foundation: organization that monitors the number of command and control servers on the Internet

- From November 2006 through May 2007, approximately 1,400 command and control servers were found active.
- The number of individual infected drones that are controlled by these 1,400 servers reportedly grew from half a million to more than 3 million from March to May 2007.

Symantec reported that it detected 6 million bot-infected computers in the second half of 2006.
Measuring Botnets

Three Distinct Phases

- Malware Collection: Collect as many bot binaries as possible
- Binary analysis via gray-box testing: Extract the features of suspicious binaries
- Longitudinal tracking: Track how bots spread and their reach

From A Multifaceted Approach to Understanding the Botnet Phenomenon, Moheeb Abu Rajab et al, IMC’06
Measuring Botnets

Darknet: Denotes an allocated but unused portion of the IP address space.
Malware Collection

- Nepenthes is a low interaction honeypot
- Nepenthes mimics the replies generated by vulnerable services in collect the first stage exploit
- Modules in nepenthes
  - Resolve DNS asynchronous
  - Emulate vulnerabilities
  - Download files – Done here by the Download Station
  - Submit the downloaded files
  - Trigger events
  - Shellcode handler
Malware Collection

- Honeynets also used along with nepenthes
- Catches exploits missed by nepenthes
- Unpatched Windows XP
- Infected honeypot compared with base to identify Botnet binary
Gateway

- Routing to different components
- Firewall: Prevent outbound attacks & self infection by honeypots
- Detect & Analyze outgoing traffic for infections in honeypot
- Several other functions
IRC Tracker (Drone)

- Use template to mimic bot
- IRC client that connects to real IRC server
- Communicate with botmaster using bot “dialect”
- Drones modified and used to act as IRC Client by the tracker to cover lot of IP address
DNS Tracker

- Bots issue DNS queries to resolve the IP addresses of their IRC servers
- List of 800,000 legitimate DNS servers (after filtering sensitive domain names (.gov, .mil, etc.) and the servers that do not answer as needed)
- Each new IRC server is checked against the cache of a DNS server in this list
  - A cache hit means at least one bot made a request for the IRC server within the TTL of the cache
  - Discovered that 85,000 servers (~11%) involved in botnet activity
Botnet Traffic Share

![Graph showing Botnet Traffic Share over time]

- **All Traffic**
- **Botnet Spreaders**

Number of incoming SYN packets (per 10 min.)

-Time scale from 04/21 to 05/13

The graphs illustrate the variation in Botnet Traffic Share over a specified period.
**DNS Tracker Results**

<table>
<thead>
<tr>
<th>TLD</th>
<th>Fraction of svrs probed</th>
<th>Percentage of all cache hits</th>
<th>Normalized hit ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>.com</td>
<td>.55</td>
<td>82%</td>
<td>29%</td>
</tr>
<tr>
<td>.net</td>
<td>.134</td>
<td>5.5%</td>
<td>8.1%</td>
</tr>
<tr>
<td>.kr</td>
<td>.015</td>
<td>3.2%</td>
<td>40%</td>
</tr>
<tr>
<td>.org</td>
<td>.037</td>
<td>2.4%</td>
<td>13%</td>
</tr>
<tr>
<td>.cn</td>
<td>.002</td>
<td>0.9%</td>
<td>95%</td>
</tr>
<tr>
<td>.ru</td>
<td>.017</td>
<td>0.6%</td>
<td>7.3%</td>
</tr>
<tr>
<td>.de</td>
<td>.016</td>
<td>0.48%</td>
<td>6%</td>
</tr>
<tr>
<td>.edu</td>
<td>.01</td>
<td>0.4%</td>
<td>8%</td>
</tr>
<tr>
<td>.ro</td>
<td>.004</td>
<td>0.32%</td>
<td>0.4%</td>
</tr>
<tr>
<td>.jp</td>
<td>.022</td>
<td>0.25%</td>
<td>2.2%</td>
</tr>
<tr>
<td>other</td>
<td>.21</td>
<td>4.45%</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Bot Scan Method

- 2 Types
  - Immediately start scanning the IP space looking for new victims after infection: 34/192 (worm-like behavior)
  - Scan when issued some command by botmaster

<table>
<thead>
<tr>
<th></th>
<th>Default Topic</th>
<th>Botmaster Command</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Localized scanning</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Class A</td>
<td>66%</td>
<td>15%</td>
</tr>
<tr>
<td>- Class B</td>
<td>11%</td>
<td>18%</td>
</tr>
<tr>
<td>- Class B</td>
<td>89%</td>
<td>82%</td>
</tr>
<tr>
<td><strong>Targeted scanning</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Class A</td>
<td>32%</td>
<td>87.4%</td>
</tr>
<tr>
<td>- Class B</td>
<td>80%</td>
<td>88%</td>
</tr>
<tr>
<td>- Class B</td>
<td>20%</td>
<td>12%</td>
</tr>
<tr>
<td><strong>Uniform scanning</strong></td>
<td>2%</td>
<td>0.3%</td>
</tr>
</tbody>
</table>
Botnet Growth - DNS

(a) Semi-Exponential
(b) Staircase
(c) Linear
Botnet Growth – IRC Tracker

(a) Semi-Exponential
(b) Staircase
(c) Linear
Botnet Online Population
Botnet Software Taxonomy

Services Launched in Victim Machine

<table>
<thead>
<tr>
<th>Utility Software Thread</th>
<th>Frequency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AV/FW Killer</td>
<td>49</td>
</tr>
<tr>
<td>Identi Server</td>
<td>43</td>
</tr>
<tr>
<td>System Security Monitor</td>
<td>40</td>
</tr>
<tr>
<td>Registry Monitor</td>
<td>38</td>
</tr>
</tbody>
</table>

OS of Exploited Host

<table>
<thead>
<tr>
<th>OS version</th>
<th>% inf.</th>
<th>Service Pack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>None</td>
</tr>
<tr>
<td>Win XP</td>
<td>82.6</td>
<td>.47</td>
</tr>
<tr>
<td>Win 2000</td>
<td>16.1</td>
<td>.09</td>
</tr>
<tr>
<td>Win Server</td>
<td>1.3</td>
<td>.57</td>
</tr>
</tbody>
</table>

![Graph showing the percentage of all captured executions for different malware families]
Botmaster Analysis

Migration between botnets (due to attack or leasing)

<table>
<thead>
<tr>
<th>Command Type</th>
<th>Frequency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>33</td>
</tr>
<tr>
<td>Scanning</td>
<td>28</td>
</tr>
<tr>
<td>Cloning</td>
<td>15</td>
</tr>
<tr>
<td>Mining</td>
<td>7</td>
</tr>
<tr>
<td>Download</td>
<td>7</td>
</tr>
<tr>
<td>Attack</td>
<td>7</td>
</tr>
<tr>
<td>Other</td>
<td>3</td>
</tr>
</tbody>
</table>
Sources

- *The Ghost in the Browser: Analysis of Web-based Malware*, Niels Provos et al., HotBots’07
- *Wide Scale Botnet Detection and Characterization*, Anestis Karasaridis et al., HotBots’07
- *The Anatomy of Clickbot.A*, Neil Daswani et al., HotBots’07
- *A Multifaceted Approach to Understanding the Botnet Phenomenon*, Moheeb Abu Rajab et al, IMC’06
  
Slides for Individual Reading from the Textbook
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

1. **Key Exchange**
   - Write the message m
   - Plaintext
   - Encryption key k
   - Encryption algorithm E
   - Cipher text
   - Insecure channel
   - Attacker
   - Decryption key k
   - Decryption algorithm D
   - Plaintext
   - Read the message m
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 pseudo-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
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 \( E(k_e, N)(m) = m^{k_e} \mod N \), where \( k_e \) satisfies \( k_e k_d \mod (p-1)(q-1) = 1 \)
- The decryption algorithm is then \( D(k_d, N)(c) = c^{k_d} \mod N \)
Asymmetric Encryption Example

- For example, make $p = 7$ and $q = 13$
- We then calculate $N = 7 \times 13 = 91$ 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 now 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

1. Write the message 69.
2. Encrypt the message 69 using the encryption key $k_{5,91}$ to get $69^5 \mod 91$.
3. Send the encrypted message $62$ over an insecure channel.
4. Decrypt the message using the decryption key $k_{29,91}$ to get $62^{29} \mod 91$.
5. Read the decrypted message 69.
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
For a message \( m \), a computer can generate an authenticator \( a \in A \) such that \( V(k)(m, a) = \text{true} \) 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 \( m' \neq m \) 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
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 $S(k_s)(m) = H(m)^{k_s} \mod N$
  - 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 $V(k_v)(m, a) \equiv (a^{k_v} \mod N = H(m))$
    - Where $k_v$ satisfies $k_v k_s \mod (p - 1)(q - 1) = 1$
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
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