Chapter 15: Security
Chapter 15: Security

- The Security Problem
- Authentication
- Program Threats
- System Threats
- Securing Systems
- Intrusion Detection
- Encryption
- Windows NT
The Security Problem

■ Security must consider **environment** of the system, and protect it from malicious access such as:
  ● Unauthorized **reading** of data (theft of information)
  ● Unauthorized **modification** or destruction of data
  ● **Preventing** legitimate use of system (denial of service)

■ **Cannot protect completely** from malicious misuse
  ● Must make too time consuming and expensive to be successful

■ **Levels that must be protected** from malicious access
  1. Physical
  2. Human
  3. Network
  4. Operating system
The Security Problem

- 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
## User Authentication

- **Major security concern** for operating systems

- 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 – age passwords
  - Use of “non-guessable” / non-trivial passwords -- rules
  - Log all invalid access attempts – and disable after $n$ tries.
  - Avoid clear text transmission (or visual monitoring) of passwords
  - Passwords may also either be encrypted or allowed to be used only once, PW file restricted / encrypted, etc.

- **Other authentication methods**
  - Paired or one-time passwords, PINs, smart cards, etc.
  - Biometrics
  - Multiple methods often safest
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

communication

receiver

attacker
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 chain
Program Threats

■ Trojan Horse
  ● Code segment that misuses its environment.
    ➢ Generic term for viruses, worms, trojans, etc.
  ● Exploits mechanisms for allowing programs written by users to be executed by other users.

■ Trap Door
  ● Specific user identifier or password that circumvents normal security procedures.
  ● Could be included in a compiler (or OS, if you are MS).

■ Stack and Buffer Overflow
  ● Exploits a bug in a program (overflow either the stack or memory buffers.)
    ➢ Get program in error state and gain additional authority
C Program with Buffer-overflow Condition

```c
#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;
    }
}
```
Typical Stack Frame Layout

- return address
- saved frame pointer
- automatic variables
- parameter(s)
```
#include <stdio.h>
int main(int argc, char *argv[])
{
    execvp("\bin\sh", \bin \sh", NULL);
    return 0;
}
```
Stack Frame Replaced With Malicious Code

(a)

- return address
- saved frame pointer
- buffer(BUFFER_SIZE - 1)
- ...
- buffer(1)
- buffer(0)

(b)

- address of modified shell code
- NO_OP
- ...
- modified shell code

Stack frame replaced with malicious code.
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

  ```vba
  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
  - Copies of itself “eat up” system resources

- **Internet worm (Robert Morris, 1988)**
  - Exploited UNIX networking features (remote access) and bugs in `finger` and `sendmail` programs.
  - Grappling hook program uploaded main worm program.

- **Viruses** – fragment of code embedded in a legitimate program.
  - Mainly effect microcomputer systems.
  - Downloading viral programs from public bulletin boards or exchanging floppy disks containing an infection.
  - Macros or scripts attached to mail
  - *Practice safe computing.*

- **Denial of Service**
  - Overload the targeted computer, preventing it from doing any useful work.
  - Attacker has not penetrated system, so little chance of damage to data
The Morris Internet Worm

The Morris Internet Worm

grappling hook

rsh attack
finger attack
sendmail attack
request for worm

worm sent

worm

target system

infected system
Threat Monitoring

- Check for suspicious patterns of activity – i.e., several incorrect password attempts may signal password guessing.

- Audit log – records the time, user, and type of all accesses to an object
  - Useful for recovery from a violation and developing better security measures.

- Scan the system periodically for security holes (“health checks”)
  - Done when the computer is relatively unused.
Threat Monitoring (Cont.)

- **Check for:**
  - Short or easy-to-guess passwords
  - Unauthorized set-uid programs
  - Unauthorized programs in system directories
  - Unexpected long-running processes
  - Improper directory protections
  - Improper protections on system data files
  - Dangerous entries in the program search path (Trojan horse)
  - Changes to system programs: monitor checksum values
Intrusion Detection

- Detect attempts to intrude into computer systems.

**Detection methods:**

- **Auditing and logging.**
- **Tripwire** (UNIX software that checks if certain files and directories have been altered – i.e. password files)
  - Similar tools available for other OSs

**System call monitoring**

- Record in log, then search for unusual patterns
<table>
<thead>
<tr>
<th>system call</th>
<th>distance = 1</th>
<th>distance = 2</th>
<th>distance = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>read</td>
<td>mmap</td>
<td>mmap close</td>
</tr>
<tr>
<td></td>
<td>getrlimit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>read</td>
<td>mmap</td>
<td>mmap</td>
<td>open</td>
</tr>
<tr>
<td>mmap</td>
<td>mmap</td>
<td>open</td>
<td>getrlimit mmap</td>
</tr>
<tr>
<td></td>
<td>open</td>
<td>close</td>
<td></td>
</tr>
<tr>
<td></td>
<td>close</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getrlimit</td>
<td>mmap</td>
<td>close</td>
<td></td>
</tr>
<tr>
<td>close</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cryptography as a Security Tool

- Broadest security tool available
  - Source and destination of messages cannot be trusted without cryptography
  - Based on secrets (keys)

- Encrypt clear text into cipher text.

- Properties of good encryption technique:
  - Relatively simple for authorized users to encrypt and decrypt data.
  - Encryption scheme depends not on the secrecy of the algorithm but on a parameter of the algorithm called the encryption key.
  - Extremely difficult for an intruder to determine the encryption key.
Secure Communication over Insecure Medium

Key exchange

write message m

encryption key k

encryption algorithm E

ciphertext c = E(k)(m)

insecure channel

attacker

decryption algorithm D

plaintext m = D(k)(c)

read 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 psuedo-random-bit generator
    - Generates an infinite **keystream**
Encryption Example - **SSL**

- **SSL** – Secure Socket Layer
  - Many options, can be complex
  - Cryptographic protocol that limits two computers to exchange messages only with each other.
  - Used between web servers and browsers for secure communication (credit card numbers)

- **Initiated by client**
  - Assumes prior to communication, server has obtained a [certificate](#) from a [certification authority](#)

- Communication between computer uses **symmetric key cryptography, plus digital certificates, random keys, timeouts**
  - See text for more details
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 known 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 $E(k_e, N)(m) = m^{k_e} \pmod{N}$, where $k_e$ satisfies $k_e k_d \pmod{(p-1)(q-1)} = 1$
  - The decryption algorithm is then $D(k_d, N)(c) = c^{k_d} \pmod{N}$
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 ciphertext 62
- Ciphertext 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

write → message 69

plaintext

encryption key \( k_{5,91} \) → \( 69^5 \mod 91 \)

insecure channel

62

decryption key \( k_{29,91} \) → \( 62^{29} \mod 91 \)

read → 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 $V : K \rightarrow (M \times A \rightarrow \{\text{true}, \text{false}\})$. 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 $V(k)(m, a) = \text{true}$ 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 \textbf{MD5}, which produces a 128-bit hash, and \textbf{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 \( S(k_s)(m) = H(m)^s \mod N \)
  - 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 \( V(k_v)(m, a) \equiv (a^v \mod N = H(m)) \)
    - 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
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
A firewall is placed between trusted and untrusted hosts.

- The firewall limits network access between these two security domains.
- Sometimes restricts communication to specific ports, IP addresses, users, processes, actions

DMZ – stand-alone domain

- Communications allowed between outside and DMZ and between DMZ and inside, but not between outside and inside
- DMZ also often includes firewalls, etc.
- May allow only certain access only to certain commands and certain systems
- Must disable many services, etc. to be secure

Can be reasonably secure with combination of measures
Network Security Through Domain Separation Via Firewall

- Internet access from company’s computers
- DMZ access from Internet
- access between DMZ and company’s computers
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.
Configurable security allows policies ranging from D to C2.

Security is based on user accounts where each user has a security ID.

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 NT 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