Chapter 15: Security

- The Security Problem
- Program Threats
- System and Network Threats
- Cryptography as a Security Tool
- User Authentication
- Firewalling to Protect Systems and Networks
- 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

- Protection is strictly an *internal* problem: How do we provide controlled access to programs and data stored in a computer system?
- Security must consider external environment of the system, and protect the system resources from:
  - unauthorized access.
  - Intentional (malicious) modification or destruction.
  - accidental introduction of inconsistency.
- Protection mechanisms work well only as long as the users conform to the intended use of and access to the system resources
- Intruders (crackers) attempt to breach security
  - Threat is potential security violation
  - Attack is attempt to breach security
- Attacks can be accidental or malicious
  - Easier to protect against accidental than malicious misuse
Security Violations

- Categories
  - **Breach of confidentiality**: unauthorized reading of data
  - **Breach of integrity**: unauthorized modification of data.
  - **Breach of availability**: unauthorized destruction of data.
    - Web-site defacement
  - **Theft of service**: unauthorized use of resources.
    - Install a daemon to act as a web server
  - **Denial of service**: Preventing legitimate use of the system

Program Threats

- Trojan Horse
  - It is a code segment that misuses its environment
  - Exploits mechanisms for allowing programs written by users to be executed by other users
    - A text-editor program, for example, may include code to search the file to be edited for certain keywords. If any are found, the entire file may be copied to a special area accessible to the creator of the text editor.
    - A program that emulates a login program
  - Spyware, pop-up browser windows, covert channels: capture information from the user's system and return it to a central site.
    - Later used to send spam → theft of service
    - All due to violation of the **principle of least privilege**.
      - a user may run with more privileges than necessary
      - an operating system may allow by default more privileges than a normal user needs
Program Threats (contd)

- Trap Door
  - The designer of a program or system might leave a hole in the software that only she is capable of using (usually to bypass some normal check)
    - Specific user identifier or password that circumvents normal security procedures
  - Could be included in a compiler
    - Generated in each code compiled!!!
  - Difficult to detect.

- Logic Bomb
  - Program that initiates a security incident under certain circumstances (date, sequence of input, external trigger, …etc)
    - a piece of code written by one of a company's (currently employed) programmers and secretly inserted into the production operating system. If, for example, the logic bomb does not get fed its daily (or weekly or monthly, etc) password, so it goes off
  - Difficult to detect.

Program Threats (Cont.)

- Viruses
  - Code fragment embedded in legitimate program
  - Very specific to CPU architecture, operating system, applications
  - most disruptive security attack
  - Usually borne via email or as a macro
    - Visual Basic Macro to reformat hard drive
      ```vbs
      Sub AutoOpen()
      Dim oFS
      Set oFS = CreateObject(’’Scripting.FileSystemObject’’)
      vs = Shell(’’c:command.com /k format c:’’,vbHide)
      End Sub
      ```
Program Threats (contd)

- Stack and Buffer Overflow
  - Most common way for an attacker outside the system, on a network or dial-up connection, to gain unauthorized access to the target system
  - Might be used also by an authorized user to gain more privileged access
  - Exploits a bug in a program (overflow either the stack or memory buffers)
    - Programmers neglect input bound checking, so the attacker sends more data than the program was expecting
  - Using trial and error (or inspecting the code), an attacker can do:
    1. Overflow an input field, command-line argument, or input buffer until it writes into the stack.
    2. Overwrite the current return address on the stack with the address of the exploit code loaded in step 1
    3. Write a simple set of code for the next space in the stack that includes the commands that the attacker wishes to execute

Modified Shell Code

- An attacker could execute a buffer overflow attack. The goal is to replace the return address in the stack frame so that it now points to the code segment containing the attacking program.
- Consider this code
  ```c
  #include <stdio.h>
  int main(int argc, char *argv[])
  {
    execvp("\bin\sh\",\"bin \sh\", NULL);
    return 0;
  }
  ```
  - This code creates a shell process using the `execvp()` system call.
  - If the program being attacked runs with system-wide permissions, this newly created shell will gain complete access to the system.
Using a debugger, the programmer then finds the address of buffer[0] in the stack.
- That address is the location of the code the attacker wants executed.
- The binary sequence of the attacker's code is appended with the necessary amount of NO-OP instructions to fill the stack frame up to the location of the return address.
- Then, the location of buffer [0], the new return address, is added.
- Finally, the attack is complete when the attacker gives this constructed binary sequence as input to the process.
- The process then copies the binary sequence from argv [1] to position buffer [0] in the stack frame.
- Now, when control returns from main (), instead of returning to the location specified by the old value of the return address, we return to the modified shell code, which runs with the access rights of the attacked process.
- One solution to this problem is for the CPU to have a feature that disallows execution of code in a stack section of memory.
System and Network Threats

- Abuse of services and network connections
- Worms – use spawn mechanism; standalone program; degrade performance
- 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
  - a means for a cracker to detect a system's vulnerabilities, then if exist to attack them.
  - Nmap: a tool that determines what services are running on a host
- Denial of Service: not stealing info, but disrupting legitimate use of a system.
  - Overload the targeted computer preventing it from doing any useful work
  - frequently easier than breaking into a machine, and generally network based.
  - Distributed denial-of-service (DDOS) come from multiple sites at once
  - Here is a one-line program that used to wipe out any UNIX system:
    ```c
    main() {while (1)fork();}
    ```

Cryptography as a Security Tool

- Broadest security tool available
  - Source and destination of network messages cannot be trusted without cryptography
    - In an isolated computer, the operating system can reliably determine the sender and recipient of all interprocess communication
  - Means to constrain potential senders (sources) and / or receivers (destinations) of messages
  - Based on secrets (keys) that is supposed to be known only by the sender and the receiver.
    - keys are designed so that it is not computationally feasible to derive them from the messages they were used to generate or from any other public information
    - Standard cryptographic practice is to assume that attacker knows the algorithm used to encipher the plaintext, but not the specific cryptographic key
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 $E: K \to (M \to C)$. 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 $D: K \to (C \to M)$. 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 (Block cipher)
    - DES works by taking a 64-bit value and a 56-bit key and performing a series of transformations to that data
  - Transformations algorithms can be known or unknown (S-box)
  - cipher-block chaining: chunks are not just encrypted but also XORed with the previous ciphertext block before encryption
- Triple-DES considered more secure (DES repeated 3 times)
  - two encryptions and one decryption, using two or three keys each time

Symmetric Encryption (contd)

- Advanced Encryption Standard (AES)
  - Replaces DES
  - Compact and efficient.
  - Uses 128, 192, 356 key lengths
- twofish
  - fast, compact, and easy to implement.
  - key length of up to 256 bits and works on 128-bit blocks
- 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: an infinite set of keys that can be used for the input plaintext stream.
Asymmetric Encryption

- Uses different encryption and decryption keys
- 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 $E(k_e, N)(m) = m^{k_e} \mod N$, 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 modulus $N = 7 \times 13 = 91$ 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

Example 2

- Step 1. Pick two large primes $p$ and $q$ (really should be 150 digits or so)
  - We pick 13 and 19. Calculate modulus $N = p \cdot q$. $N = 13 \times 19 = 247$
- Step 2. Find $e$ so that $e$ is relatively prime to $(p-1)(q-1)$.
  - For the above $p$ and $q$, $(p-1)(q-1) = 12 \times 18 = 216$. I choose $e = 11$.
    - $(e, N)$ is the public key. So I let everyone know my public key is $(11, 247)$
- Step 3. Find private key. Find a number $d$ so the $e \cdot d = 1 \mod (p-1)(q-1)$. this means that $11 \cdot d = 1 \mod 216$. It turns out that $d = 59$ works.
  - $(11 \times 59 = 649$ and $649 \div 216$ is $3$ with remainder $1$)
  - I keep my $d = 59$ secret!
Example 2 (contd)

- Someone else wants to send me a message. The message is changed into a number (change to ASCII in hex, then compute in decimal), say \( m = 102 \).
- The sender encrypts \( m \) by the formula \( c = m^e \mod N \). For this message, using my public key we need to compute \( c = 102^{11} \mod 247 \)
- Computation:
  - \( 102^1 = 102 \mod 247 \)
  - \( 102^2 = 30 \mod 247 \)
  - \( 102^4 = 30^2 = 159 \mod 247 \)
  - \( 102^8 = 159^2 = 87 \mod 247 \)
  - \( 102^{11} = 102^8 \times 102^2 \times 102^1 = 87 \times 30 \times 102 = 201 \mod 247 \)
- The encrypted message sent to me is \( c = 201 \).

Example 2 (contd)

- Reading my message
- Use the formula \( m = c^d \mod N \). I need to find \( m = 201^{59} \mod 247 \)
- Computation:
  - \( 201^1 = 201 \mod 247 \)
  - \( 201^2 = 140 \mod 247 \)
  - \( 201^4 = 140^2 = 87 \mod 247 \)
  - \( 201^8 = 87^2 = 159 \mod 247 \)
  - \( 201^{16} = 159^2 = 87 \mod 247 \)
  - \( 201^{32} = 87^2 = 159 \mod 247 \)
  - \( 201^{59} = 201^{32} \times 201^{16} \times 201^8 \times 201^2 \times 201^1 \)
    - \( = 159 \times 87 \times 159 \times 140 \times 201 \mod 247 \)
    - \( = 102 \mod 247 \), so my message is 102
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
  - Used for encryption of small amounts of data, authentication, confidentiality, and key distribution
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
- Message digests are useful for detecting changed messages but are not useful as authenticators
- \( H(m) \) can be sent along with a message; but if \( H \) is known, then someone could modify \( m \) and recompute \( H(m) \), and the message modification would not be detected.
- Therefore, an authentication algorithm takes the message digest and encrypts it.

Authentication Algorithms - MAC

- Based on Symmetric encryption used in authentication algorithm
- a cryptographic checksum (message-authentication code “MAC”) is generated from the message using a secret key.
- 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.)

- Encryption can prove the identity of the sender of a message, then why do we need separate authentication algorithms
  - Fewer computations (except for RSA digital signatures)
  - Authenticator usually shorter than message
    - improves space use and transmission time efficiency.
  - Sometimes want authentication but not confidentiality
    - Signed patches et al
  - Can be basis for non-repudiation
    - Nonrepudiation assures that a person filling out an electronic form, for example, cannot deny that he did so.
Key Distribution

- Delivery of symmetric key is huge challenge
  - Sometimes done out-of-band
    - do not scale well
- Asymmetric keys can proliferate – stored on key ring
  - keys be exchanged in public
  - One private key only for a user
  - Even asymmetric key distribution needs care – man-in-the-middle attack

Man-in-the-middle Attack on Asymmetric Cryptography
Digital Certificates

- We need proof of who or what owns a public key, to mitigate the man-in-the-middle-attack ⇒ use a digital certificate
- It is a public key digitally signed by 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 then 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 (card) and/or a user attribute (fingerprint)
  - A password can be associated with each object or user.
- Passwords must be kept secret
  - Frequent change of passwords
  - Use of “non-guessable” passwords (can’t help in brute force)
  - Log all invalid access attempts
  - Don’t write it down!!! (especially if it is hard to remember)
  - Don’t share your ID with someone else
- Passwords may also either be encrypted or allowed to be used only once

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
- Vulnerabilities: can be tunneled or spoofed
  - Tunneling allows “disallowed protocol” to travel within allowed protocol (i.e. telnet inside of HTTP)
  - Spoofed: an unauthorized host pretends to be an authorized host by meeting some authorization criterion.
  - 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

- A network firewall can separate a network into multiple domains
  - Un-trusted domain;
  - DMZ
  - Company's computers
- Connections are allowed from the Internet to the DMZ computers
- And, from the company computers to the Internet
- But are not allowed from the Internet or DMZ computers to the company computers.

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