Security

- What's the problem that security tries to solve?
- Authentication
- Encryption
- Threats to the operating system
  - » Program threats
  - » System threats
  - » Intrusion detection

The Security Problem

- Computer systems must be protected from external threats
  - » Unauthorized access
  - » Malicious modification and/or destruction
  - » Accidental inconsistencies
- Fortunately, easier to protect against accidents than malicious intruders
  - » Accidents aren't often repeated
  - » Intruders may try to exploit several weaknesses at once, not likely to happen by accident

Authentication

- System needs to determine that an external entity is what it claims to be
  - » Human users
  - » Other computers
- Often accomplished by passwords
  - » Only actual entity would know the password
  - » Passwords must be difficult to guess
  - » Passwords should be changed often (or they might be discovered)
  - » Invalid login attempts should be logged to track people trying to log in "illegally"

Dealing With Passwords

- Passwords should be memorable
  - » Users shouldn't need to write them down!
  - » Users should be able to recall them easily
- Passwords shouldn't be stored "in the clear"
  - » Password file is often readable by all system users!
  - » Password must be checked against entry in this file
- Solution: use hashing to hide "real" password
  - » One-way function converting password to meaningless string of digits (Unix password hash, MD5)
  - » Difficult to find another password that hashes to the same meaningless string
  - » Knowing the hashed value and hash function gives no clue to the original password

Password Issues

- Passwords can be guessed
  - » Hackers can get a copy of the password file
  - » Run through dictionary words and names
    - – Hash each name
    - – Look for a match in the file
- Solution: use “salt”
  - » Random characters added to the password before hashing
  - » Salt characters stored “in the clear”
  - » Increase the number of possible hash values for a given password
    - – Actual password is “pass”
    - – Salt = “aa” => hash “passaa”
    - – Salt = “bb” => hash “passbb”
  - » Result: cracker has to try many more combinations

What’s Encryption Good For?

- Authentication is usually one-way: once hashed, the original text can’t be recovered
- Encryption is two-way: the original text can be recovered from the encrypted text by decryption
- Encryption can be used to:
  - » Protect information on a computer system from seeing private information
  - » Protect information in transit between computers
  - » Provide proof that a message was sent by a particular entity: if encrypted by a key that only A & B know, A knows that B must have sent the message

Encryption Basics

- Algorithms consist of
  - » Encryption algorithm E & decryption algorithm D
  - » Encryption & decryption keys (may be the same)
  - » $D(k_d,E(k_e,m)) = m$
- Good encryption techniques
  - » Allow simple encryption & decryption by users
  - » Rely not on the secrecy of the algorithm but instead on the secrecy of the *encryption key*
  - » Prevent intruders from easily finding the encryption key
  - » Can use variable length keys; longer keys are harder to crack
- Two basic kinds of encryption
  - » Shared-secret (single key): single key (or set of keys) known to both parties
  - » Public key encryption: half of key is made public, and the other half is kept private

Shared Secret Encryption

- Sender and receiver share keys
  - » Key distribution method must be secure…
  - » Key must be large and difficult to guess
- Doubling key size may make code twice as hard to crack
- Example: Data Encryption Standard (DES)
  - » Uses 56-bit keys
  - » Encrypts data a block at a time
  - » Same key is used to encrypt & decrypt
  - » Keys used to be difficult to guess
    - – Needed to try 2^{55} different keys, on average
    - – Modern computers can try millions of keys per second with special hardware
    - – For $250K, EFF built a machine that broke DES quickly

Shared Secret: One-Time Pad

- For unbreakable communication, use one-time pad
  - » Truly random key as long as message
  - » XOR bits of key with bits of message
- Code is unbreakable because
  - » Key could be anything
  - » Without knowing key, message could be anything with the correct number of bits in it
- Difficulty: distributing key, often as hard as distributing message
- Difficulty: generating truly random bits
  - » Can’t use computer random number generator!
  - » May use physical processes (radioactive decay, lava lamp…)

Public Key Encryption

- Instead of using a single shared secret, keys come in pairs
  - » One key of each pair distributed widely (*public key*), k_p
  - » One key of each pair kept secret (*private or secret key*), k_s
  - » Two keys are inverses of one another, but not identical
  - » Encryption & decryption are the same algorithm, so $E(k_p,E(k_s,m) = E(k_s,E(k_p,m) = m$
- Currently, most popular method involves primes and exponentiation
  - » Difficult to crack unless large numbers can be factored
  - » Very slow for large messages

More on Public Key Encryption

- Public, private key pair consists of $k_p = (d,n)$ & $k_s = (e,n)$
  - » $n = p \times q$
  - » d is a randomly chosen integer with GCD $(d, (p-1) \times (q-1)) = 1$
  - » e is an integer such that $(e \times d)$ MOD $(p-1) \times (q-1) = 1$
- p & q aren’t published, and it’s hard to find them: factoring large numbers is thought to be NP-hard
- Public key is published, and can be used by anyone to send a message to the private key’s owner
- Encryption & decryption are the same algorithm: $E(k,m) = m^k$ MOD n
  - » Methods exist for doing the above calculation quickly, but...
  - » Exponentiation is very slow
  - » Public key encryption not usually done with large messages

Pretty Good Privacy (PGP)

- Uses public key encryption
  - » Facilitates key distribution
  - » Allows messages to be sent encrypted to a person (encrypt with person’s public key)
  - » Allows person to send message that must have come from her (encrypt with person’s private key)
- Problem: public key encryption is very slow
- Solution: use public key encryption to exchange a shared key
  - » Shared key is relatively short (~1024 bits)
  - » Message encrypted using shared key encryption
- PGP can also be used to authenticate sender
  - » Use one-way hash on message, leave message unencrypted
  - » Encrypt the hash with user’s private key
  - » Anyone can prove that user sent the message

Program Threats

- One type of threat is from programs that a user runs that do unexpected things
  - » Trojan horse
  - » Trap door
- Trojan horse
  - » Program that looks like it does something useful
  - » Program actually does something harmful
  - » Example: put a file called "ls" into someone's home directory which actually deletes all of their files
- Trap door
  - » Program that really does its job, but…
  - » Program has a "security hole" that allows a particular user additional privileges
  - » Example: author of "login" program makes it so that the account name "steve" can log in without any password

© 1999 by Ethan L. Miller

New Kinds of Program Threats

- Old way: computers only ran programs from local, trusted sources
  - » Local disk
  - » Files available on local file server
- New way: computers run programs downloaded from random places on the Web
  - » Shareware / freeware for your PC
  - » Java bytecode files to do almost anything
- How can these programs be trusted?
  - » Don't let them use (and especially modify) any crucial data
  - » Have them signed by an authority you trust, and use cryptographic techniques to ensure the signature is genuine

© 1999 by Ethan L. Miller

System Threats

- Unlike program threats, system threats without requiring a user to explicitly start them up
  - » Viruses
  - » Worms
- Viruses
  - » Code fragments that exploit system weaknesses to run their own code instead of "normal" code in a program
    - – Code replaces the normal instructions in a program
    - – Most common in PC-type computers
  - » Systems may be "infected" from many sources
    - – Software loaded from disk
    - – E-mail & other network traffic
  - » Viruses may be benign or malevolent

© 1999 by Ethan L. Miller

Worms

- Self-replicating independent code (virus::worm == virus::bacteria)
  - » Worm installs itself via weakness in system
    - – Bug in networking code
    - – Electronic mail that's actively decoded
  - » Worm runs on its own as a "regular" process
- Famous example: Internet worm (1988)
  - » Used vulnerability in finger & sendmail to insert its code into specific types of Unix OS
  - » Sucked up CPU time on infected machines
  - » Nearly brought the Internet to its knees

© 1999 by Ethan L. Miller

Monitoring for Threats

- Check for patterns of suspicious (but normal) activities
  - » Passwords mistyped
  - » Excessive connections to the computer from somewhere
- Use logs of references to objects or services
  - » Check ftp or Web log for outgoing file names & destinations
  - » Check login log for suspicious logins (middle of the night, user was out of town, etc.)
- Scan files & programs for evidence of change
  - » Check modification dates
  - » Compare against unchangeable files (on CD-ROM, perhaps)
  - » Compare digital signature to value stored off-line to see if the file has changed

Specific Things to Monitor

- Web, ftp, and mail logs (for unusual activity)
- Login attempts (successful & unsuccessful)
- Connections to the computer (successful & unsuccessful)
- Long-running processes
- Improperly protected files & directories
- Modified or added files in system directories
  - » Look for invisible or otherwise innocuous files
  - » Check for programs that may have been replaced
- Changes in the program search path (especially for root)

Why Is Security So Difficult?

- Security is hard because
  - » Systems are extraordinarily complex
  - » System administrators can be lazy
    - – Configuration files may, by default, allow lax security
    - – Bug fixes may not be applied when they're available
  - » Attackers are both smart and persistent
    - – Try multiple methods in combination
    - – Try methods out on local machines first
- Security is an ongoing battle between the offense (people trying to break in) and defense (those defending computer systems)