Security

Announcements

• Reading
  - Chapter 15
Who do you trust?

- Do I trust a login prompt?
- Do I trust the OS that I got from the vendor?
- Do I trust the system staff?
  - should I encrypt all my files?
- Networking
  - do you trust the network provider?
  - do you trust the phone company?
- How do you bootstrap security?
  - need one “out of band” transfer to get going

The Security Problem

- Security must consider external environment of the system, and protect it from:
  - unauthorized access (confidentiality)
  - modification or destruction of data (integrity)
  - denial of service (availability)
- Easier to protect against accidental than malicious misuse
Authentication

- When an operation is performed, the computer must know which policy to check to authorize it
- Policies are based on domains
- Thus we need a way to associate a process with a domain
  - UNIX: Bootstrap for the “initial process” and then give the same user id to all sub processes

Passwords

- 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.
  - Use of “non-guessable” passwords
  - Log all invalid access attempts
Example (UNIX passwords)

- use a function that is hard to invert
  - “easy” to compute f(x) given x
  - hard to compute x given f(x)
  - the function used (crypt) is a variation of DES
    - changes selected items in the transformation matrix to prevent hardware attacks
  - store only f(x) in the filesystem
- to login
  - user supplies a password x’
  - compute f(x’) and compare to f(x)
- salt
  - add an extra two characters to x so that the same x will produce different values on different machines
- dictionary attack
  - if it’s too easy to compute f(x), can “guess” many passwords and try them out

Other authenticators

- Biometric data
- One-time passwords
  - Password function
    - checked with challenge/response
  - On-time pad
    - password, once used, is discarded
    - pad obtained from trusted source
Computer Authentication

• How does a user know what computer they are using?

• Need to have *mutual authentication*
  - computer presents some information that only it could contain
  - example: Windows <ctrl>-<alt>-<del> to login
    • user software can’t trap that information
    • assumes that the kernel itself is secure

• telephone example
  - never give banking/credit card info over the phone unless you placed the phone call
    • i.e. you use the telco namespace for authentication

Program Threats

• Trojan Horse
  - Trap door

• Abstraction-violating attack
  - Stack smashing

• Virus
• Worm
• Denial of Service Attacks
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!

- Trap Door
  - Specific user identifier or password that circumvents normal security procedures
  - Could be included in a compiler

- Stack smash via Buffer Overflow

Example attack in C

```c
#include <stdio>

int login() {
    char user [100];
    printf("login: ");
    scanf("%s", &user);
    ... // get password etc.
}
```

What happens if the user types in something that’s more than 100 characters?
Calling scanf()

Stack grows downward

32 bits

int login() {
    char user [100];
    printf("login: ");
    scanf("%s", &user);
}

Buffer indexes proceed upward

User types login

int login() {
    char user [100];
    printf("login: ");
    scanf("%s", &user);
}
int login() {
    char user [100];
    printf("login: ");
    scanf("%s", &user);
}

Can make the return address point anywhere, including into the buffer.
Abstraction-violating Attack

- Language and library abstractions may not be enforced
  - Array accesses, pointer dereferences, type casts, format strings “trusted” by the compiler
- Other attacks exploit this fact
  - Heap-based buffer overruns
  - Format string attacks

Program Threats

- Viruses - fragment of code embedded in a legitimate program. (Popularly speaking, worm and virus tend to be used synonymously.)
  - Located in acquired program; set in motion when the program is run
  - Can also infect boot sector; even harder to spot
  - Possible to write system-independent viruses
    - MS Word virus uses macros to call into the OS
Worms

- Intrusion-based DOS
  - Standalone program
  - Self-propagates

- Internet worm
  - Exploited UNIX networking features (remote access) and bugs in *finger* and *sendmail* programs
  - Grappling hook program uploaded main worm program

The Morris Internet Worm
Denial of Service Attacks

• Overload the targeted computer preventing it from doing any useful work.

• Ex: Network based
  - TCP/IP SYN attack
  - DDOS of Yahoo
    • Make use of infected zombie machines

• Ex: `fork()` bomb

Defenses

• Intrusion Detection
  - Use system logs to discover an attack post-mortem or while it’s happening
  - Discover an attempted attack before it happens

• Firewalls
  - Prevent access to risky services

• Encryption
  - Protect communications and storage
Intrusion Detection

- Detect attempts to intrude into computer systems
  - Anomaly detection: know what “normal” behavior is and look for anomalies
    - The “ls” process should only use filesystem and I/O-related system calls (i.e., never fork new programs)
  - Signature detection: look for “bad” behavior
    - Several incorrect password attempts may signal password guessing
- Behavior is determined by auditing and logging
  - Log messages
    - records the time, user, and type of all accesses to an object; useful for detection and forensics
  - Network traffic
  - System call sequences

Threat monitoring examples

- Check for:
  - Unsafe services being run (port scans)
  - 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
Tripwire

- Compute a set of expectations about system
  - Hash of file contents
  - Dates on files
- Store database of values
  - On read-only media
  - Offline
- Periodically
  - Compare database to current system
  - Report any differences

Firewall

- A firewall is placed between trusted and untrusted hosts
- The firewall limits network access between these two security domains
Network Security Through Domain Separation Via Firewall

Encryption

- For protecting data confidentiality
- Encrypt clear text into cipher text
- Properties of good encryption technique:
  - Relatively simple for authorized users to incrypt 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.
Encryption: protecting info from being read

- Given a message m
  - use a key k, and function $E_k$ to compute $E_k(m)$
  - store or send only $E_k(m)$
  - use a second key $k'$ and function $D_{k'}$ such that
    - $D_{k'}(E_k(m)) = m$
  - $E$ and $D$ need not be kept a secret

Encryption Techniques

- For $D_{k'}(E_k(m)) = m$:
  - If $k = k'$ it’s called symmetric key encryption
    - need to keep k secret
    - example Data Encryption Standard (DES)
  - if $k \neq k'$, it’s called public key encryption
    - By keeping $k'$ secret, anyone can send a private message using $k$
    - still need a way to authenticate $k$ or $k'$ for a user
    - example RSA, ElGamal
Data Encryption Standard

- *Data Encryption Standard* (DES) substitutes characters and rearranges their order on the basis of an encryption key provided to authorized users via a secure mechanism. Scheme only as secure as the mechanism.
- New standard: AES Rijndahl.
  - Developed through public competition
- Others: twofish, RC4, RC5

One Time Pad

- Key Idea: randomness in key
- Create a random string as long as the message
  - each party has the pad
  - xor each bit of the message with the a bit of the key
- Almost impossible to break
- Some practical problems
  - need to ensure key is not captured
  - a one bit drop will corrupt the rest of the message
Public Key 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 decrypt the messages.
  - Efficient algorithm for testing whether or not a number is prime.
  - No efficient algorithm is know for finding the prime factors of a number.

RSA

• RSA (Rivest-Shamir-Adelman) first public key system.
  - algorithm for computing public/private key pairs
  - based on problems involved in factoring large primes; for a message P
    • \( C = (P^e \mod N) \), and \( P = (C^d \mod N) \)
    • \( N \) is the product of two large prime numbers \( p \) and \( q \)
Message Authentication Schemes

- Use a digital signature to ensure authenticity
  - Does not require signed document to be encrypted
- Given a message m
  - use a key $k$, and function $S_k$ to compute $S_k(m)$
  - send $(m, S_k(m))$
  - use a second key $k'$ and function $V_{k'}$ such that
    - $V_{k'}(m, S_k(m)) = \text{true or false}$
  - $S$ and $V$ need not be kept a secret
- Most encryption schemes support MAC

Comparison

- Public key crypto is useful for identity
  - Associate a public key with a person. Only the holder of the corresponding private key (i.e., the person) can decrypt
    - But how to establish the mapping? Requires public key distribution scheme ...
  - Tends to be slow
- Symmetric key crypto is useful for performance
  - Can use public keys or other mechanism to establish identity, and then generate a shared secret key
Secure Communications - SSL

- SSL - Secure Socket Layer
- Cryptographic protocol that limits two computers to only exchange messages with each other
- Used between web servers and browsers for secure communication (credit card numbers)
- The server is verified with a certificate
- Communication between each computers uses symmetric key cryptography

SSL handshake

- Client requests connection with server
  - Provides a random number nc
- Server responds to client, providing
  - Certificate
    - Contains public key, other attributes
    - Signed by Certificate Authority (Verisign)
  - Random number ns
- Client verifies the signature on the certificate
SSL handshake

- Client generates pre-master secret (pms), sends that to server encrypted with server’s public key
- Server decrypts message
- Both client and server now know ns, nc, and pms, and calculate f(nc, ns, pms): the shared secret
- Shared secret used to encrypt future communications

Why is security hard?

- It’s not a feature
  - It’s about what should never happen, not about what should happen
- How do I know whether I have “enough?”
  - Testing correct functionality orthogonal to testing for resistance to attack
- Attackers constantly trying to exploit assumptions of computer system builders
  - Need well-defined semantics
    - Try to systematically demonstrate that systems (policies and programs that implement them) meet their security goals