

CS 140: Operating Systems Lecture 22: Security

Mendel Rosenblum

Encryption

- ◆ Basic idea: store and transmit information in an encoded form that doesn't make any sense.
- ◆ The basic mechanism:
 - Start with text to be protected. Initial readable text is called *clear text*.
 - Encrypt the clear text so that it doesn't make any sense at all. The nonsense text is called *cipher text*. The encryption is controlled by a secret password or number; this is called the *encryption key*. The encrypted text can be stored in a readable file, or transmitted over unprotected channels.
 - To make sense of the cipher text, it must be *decrypted* back into clear text. This is done with some other algorithm that uses another secret password or number, called the *decryption key*.

Requirements

- ◆ The encryption function cannot easily be inverted (cannot get back to clear text unless you know the decryption key).
- ◆ The encryption and decryption must be done in some safe place so the clear text can't be stolen.
- ◆ The keys must be protected. In most systems, can compute one key from the other (usually the encryption and decryption keys are identical), so can't afford to let either key leak out.

Secure communication

- ◆ Problem: how to establish secure channel (i.e. distribute keys) in the first place?
- ◆ Public key encryption: mechanism for encryption where knowing the encryption key doesn't help you to find decryption key, or vice versa.
 - User provides a single password, system uses it to generate two keys (use a one-way function, so can't derive password from either key).
 - In these systems, keys are inverses of each other: could just as easily encrypt with decryption key and then use encryption key to recover clear text.
 - Each user keeps one key secret, publicizes the other. Can't derive private key from public. Public keys are made available to everyone, in a phone book for example.

Safe Mail - Secrecy

- ◆ Use public key of destination user to encrypt mail. Anybody can encrypt mail for this user and be certain that only the user will be able to decipher it.
- ◆ It's a nice scheme because the user only has to remember one key, and all senders can use the same key. However, how does receiver know for sure who it's getting mail from?

Integrity

- ◆ Positive identification: can also use public keys to certify identity:
 - To certify your identity, use your private key to encrypt a text message, e.g. "I agree to pay Mary Wallace \$100 per year for the duration of life."
 - You can give the encrypted message to anybody, and they can certify that it came from you by seeing if it decrypts with your public key. Anything that decrypts into readable text with your public key must have come from you! This can be made legally binding as a form of *electronic signature*.
- ◆ These two forms of encryption can be combined together. To identify sender in secure mail, encrypt first with your private key, then with receiver's public key.

Encryption limitations

- ◆ Encryption appears to be a great way to thwart listeners and do authentication. It doesn't help with Trojan Horses, though.
- ◆ General problem: how do we know that an encryption mechanism is safe? It's extremely hard to prove. This is a hot topic for research: theorists are trying to find provably hard problems, and use them for proving safety of encryption.
- ◆ How safe is encryption?
DES. RC5 are 56 bit encryption systems. Why???

Improving encryption safety

- ◆ Remove "known patterns" from the clear text
Example: "Dear Sir" or your name.
Compress clear text before encryption.
- ◆ Do not send large amounts of information with the same key.
Change keys frequently.
Example: sending an 100 Megabyte file.
- ◆ Need to upgrade encryption as computers become more powerful.
Distributed.net + EFF = 22 hours to break DES, Jan 1999.

Cryptographically secure checksums

- ◆ Problem: Someone sends me a binary. How do I know if it was modified in transit?
Need a checksum check.
- ◆ Cryptographically secure checksum.
F(File) = big integer.
Also called Message Digests or Digital fingerprint.
Example: MD5/SHA1
- ◆ What if companies built PC/workstations that would only run programs with specified message digests?
Use message digests to build up environment of trust.
Makes Trojan Horses more difficult.
= "Trusted" computing.

Issues with protection mechanisms

- ◆ Keep the mechanism secret
Makes it harder to break in.
Harder to change code if secret is released.
- ◆ Publish the mechanism.
Encourage bad guy -- good guy fight.
This was done in Unix.
- ◆ Diebold voting machines?
Secret system.
Code ended up on website.
Turned out the security was a joke, and still is.

Encryption Today

Trend: Apply encryption technology to:
Communication:
Establish safe communication over a network.
Authentication:
Establish identity, certify source of info.

Key distribution is key problem

- ◆ Problem: Public key doesn't make distribution that much easier.
Example: You hear: "I, Mendel, say my public key is K"
Who said this?
- ◆ Solution:
- ◆ Trusted server: Authentication server
Trusted by everyone.
Avoids having N^2 keys around the system.
- ◆ Trusted computing base (TCB)
Software and hardware that must behave correctly.
Fail-secure.
Secure in the face of failures.

Key distribution with a trusted server

- 1) A asks a trusted server (AS) for a key to talk with B.
No encryption needed here.
- 2) AS replies with new conversation key CK.
Message uses A's key so only A can read it.
It contains the key, and the key encrypted with B's key.
- 3) A sends message to B telling it the key.
Note that A can't read the content of the message because it is encrypted with B keys.

Real-world usage: Kerberos.

Some dangers of key distribution

- ◆ Eavesdropper:
Encryption solves this one.
- ◆ Replaying:
Replay old message to confuse or fool parties. May fool into divulging information
Solution: *Nonce*
Used-once identifiers