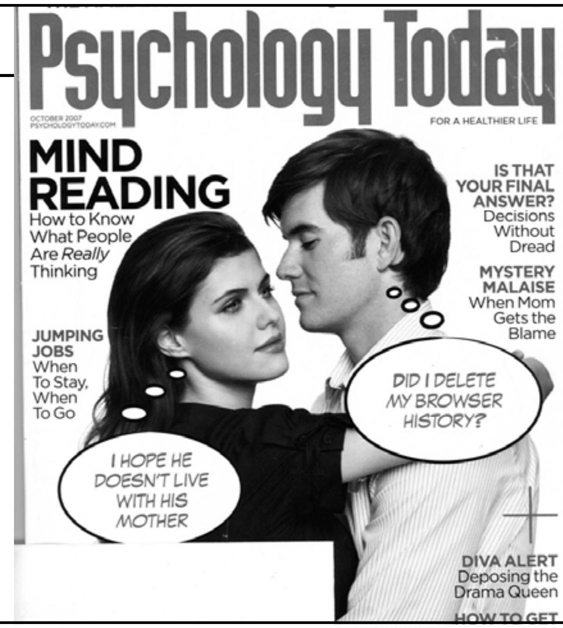
Protection and Security

How to be a paranoid
or just think like one



Leaking information

- Stealing 26.5 million veteran's data
- Data on laptop stolen from employee's home (5/06)
  - Veterans' names
  - Social Security numbers
  - Dates of birth
- Exposure to identity theft

- CardSystems exposes data of 40 million cards (2005)
  - Data on 70,000 cards downloaded from ftp server

These are attacks on privacy (confidentiality, anonymity)

The Sony rootkit

- "Protected" albums included
  - Billie Holiday
  - Louis Armstrong
  - Switchfoot
  - The Dead 60's
  - Flatt & Scruggs, etc.
- Rootkits modify files to infiltrate & hide
  - System configuration files
  - Drivers (executable files)

The Sony rootkit

- Sony's rootkit enforced DRM but exposed computer
  - CDs recalled
  - Classified as spyware by anti-virus software
  - Rootkit removal software distrubuted
  - Removal software had exposure vulnerability
  - New removal software distrubuted
- Sony sued by
  - Texas
  - New York
  - California

This is an attack on integrity

The Problem

- Types of misuse
  - Accidental
  - Intentional (malicious)
- Protection and security objective
  - Protect against/prevent misuse
- Three key components:
  - Authentication: Verify user identity
  - Integrity: Data has not been written by unauthorized entity
  - Privacy: Data has not been read by unauthorized entity

Have you used an anonymizing service?

1. Yes, for email
2. Yes, for web browsing
3. Yes, for something else
4. No

What are your security goals?

- Authentication
  - User is who s/he says they are.
  - Example: Certificate authority (verisign)
- Integrity
  - Adversary can not change contents of message
  - But not necessarily private (public key)
  - Example: secure checksum
- Privacy (confidentiality)
  - Adversary can not read your message
  - If adversary eventually breaks your system can they decode all stored communication?
  - Example: Anonymous remailer (how to reply?)
- Authorization, repudiation (or non-repudiation), forward security (crack now, not crack future), backward security (crack now, not cracked past)

What About Security in Distributed Systems?

- Three challenges
  - Authentication
    - Verify user identity
  - Integrity
    - Verify that the communication has not been tempered with
  - Privacy
    - Protect access to communication across hosts
- Solution: Encryption
  - Achieves all these goals
  - Transform data that can easily reversed given the correct key (and hard to reverse without the key)
- Two common approaches
  - Private key encryption
  - Public key encryption
- Cryptographic hash
  - Hash is a fixed sized byte string which represents arbitrary length data. Hard to find two messages with same hash.
  - If m != m' then H(m) != H(m') with high probability. H(m) is 256 bits

Private Key (Symmetric Key) Encryption

- Basic idea:
  - {Plain text}^K → cipher text
  - {Cipher text}^K → plain text
  - As long as key K stays secret, we get authentication, secrecy and integrity
- Infrastructure: Authentication server (example: kerberos)
  - Maintains a list of passwords; provides a key for two parties to communicate
- Basic steps (using secure server S)
  - A → S {Hi! I would like a key for AB}
  - S → A {Use Kab {This is A! Use Kab}^Kb}^Ka
  - A→ B {This is A! Use Kab}^Kb
  - Master keys (Ka and Kb) distributed out-of-band and stored securely at clients (the bootstrap problem)
- Refinements
  - Generate temporary keys to communicate between clients and authentication server

Public Key Encryption

- Basic idea:
  - Separate authentication from secrecy
  - Each key is a pair: K-public and K-private
  - {Plain text}^K-private → cipher text
  - {Cipher text}^K-public → plain text
  - K-private is kept a secret; K-public is distributed
- Examples:
  - {I'm Emmett}^K-private
    - Everyone can read it, but only I can send it (authentication)
  - {Hi, Emmett}^K-public
    - Anyone can send it but only I can read it (secrecy)
- Two-party communication
  - A → B {I'm A {use Kab}^K-privateA}^K-publicB
  - No need for an authentication server
  - Question: how do you trust the "public key" server?
    - Trusted server: {K-publicA}^K-privateS

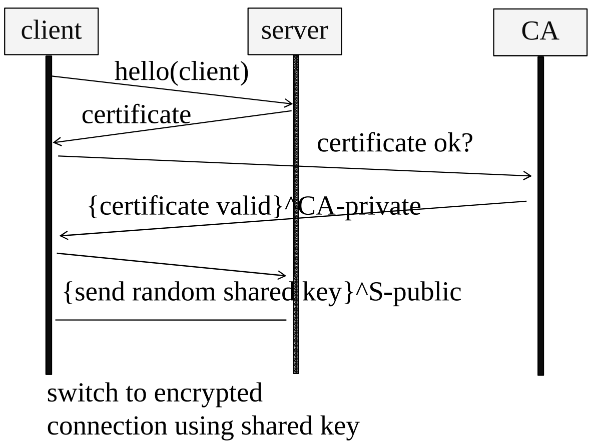
Implementing your security goals

- Authentication
  - {I'm Emmett}^K-private
- Integrity
  - {SHA-256 hash of message I just send is …}^K-private
- Privacy (confidentiality)
  - Public keys to exchange a secret
  - Use shared-key cryptography (for speed)
  - Strategy used by ssh
- Forward/backward security
  - Rotate shared keys every hour
- Repudiation
  - Public list of cracked keys

When you log into a website using an http URL, which property are you missing?

1. Authentication
2. Integrity
3. Privacy
4. Authorization
5. None

Securing HTTP: HTTPS (HTTP+SSL/TLS)



When you visit a website using an https URL, which property are you missing?

1. Authentication (server to user)
2. Authentication (user to server)
3. Integrity
4. Privacy
5. None

Authentication

- Objective: Verify user identity
- Common approach:
  - Passwords: shared secret between two parties
  - Present password to verify identity

1. How can the system maintain a copy of passwords?
   - Encryption: Transformation that is difficult to reverse without right key
   - Example: Unix /etc/passwd file contains encrypted passwords
   - When you type password, system encrypts it and then compared encrypted versions

Authentication (Cont'd.)

2. Passwords must be long and obscure
   - Paradox:
     - Short passwords are easy to crack
     - Long passwords – users write down to remember ➔ vulnerable
   - Original Unix:
     - 5 letter, lower case password
     - Exhaustive search requires 26^5 = 12 million comparisons
     - Today: < 1us to compare a password ➔ 12 seconds to crack a password
   - Choice of passwords
     - English words: Shakespeare's vocabulary: 30K words
     - All English words, fictional characters, place names, words reversed, … still too few words
     - (Partial) solution: More complex passwords
       - At least 8 characters long, with upper/lower case, numbers, and special characters

Are Long Passwords Sufficient?

- Example: Tenex system (1970s – BBN)
  - Considered to be a very secure system
  - Code for password check:

```
For (i=0, i<8, i++) {
        if (userPasswd[i] != realPasswd[i])
        Report Error;
}
```

  - Looks innocuous – need to try 256^8 (= 1.8E+19) combinations to crack a password
  - Is this good enough??

No!!!

Are Long Passwords Sufficient? (Cont'd.)

- Problem:
  - Can exploit the interaction with virtual memory to crack passwords!
- Key idea:
  - Force page faults at carefully designed times to reveal password
  - Approach
    - Arrange first character in string to be the last character in a page
    - Arrange that the page with the first character is in memory
    - Rest is on disk (e.g., a|bcdefgh)
    - Check how long does a password check take?
      - If fast → first character is wrong
      - If slow → first character is right → page fault → one of the later character is wrong
    - Try all first characters until the password check takes long
    - Repeat with two characters in memory, …
  - Number of checks required = 256 * 8 = 2048 !!
- Fix:
  - Don't report error until you have checked all characters!
  - But, how do you figure this out in advance??
  - Timing bugs are REALLY hard to avoid

Alternatives/enhancements to Passwords

- Easier to remember passwords (visual recognition)
- Two-factor authentication
  - Password and some other channel, e.g., physical device with key that changes every minute
  - http://www.schneier.com/essay-083.html
  - What about a fake bank web site? (man in the middle)
  - Local Trojan program records second factor
- Biometrics
  - Fingerprint, retinal scan
  - What if I have a cut? What if someone wants my finger?
- Facial recognition

Password security

- Instead of hashing your password, I will hash your password concatenated with a random salt. Then I store the unhashed salt along with the hash.
  - (password . salt)^H salt
- What attack does this address?

1. Brute force password guessing for all accounts.
2. Brute force password guessing for one account.
3. Trojan horse password value
4. Man-in-the-middle attack when user gives password at login prompt.

Authorization

- Objective:
  - Specify access rights: who can do what?
- Access control: formalize all permissions in the system

	File1	File2	File3	...
User A	RW	R	--	...
User B	--	RW	RW	..
User C	RW	RW	RW	...

- Problem:
  - Potentially huge number of users, objects that dynamically change ➔ impractical
- Access control lists
  - Store permissions for all users with objects
  - Unix approach: three categories of access rights (owner, group, world)
  - Recent systems: more flexible with respect to group creation
- Privileged user (becomes security hole)
  - Administrator in windows, root in Unix
  - Principle of least privlege

Authorization

- Capability lists (a capability is like a ticket)
  - Each process stores information about objects it has permission to touch
  - Processes present capability to objects to access (e.g., file descriptor)
  - Lots of capability-based systems built in the past but idea out of favor today

Enforcement

- Objectives:
  - Check password, enforce access control
- General approach
  - Separation between "user" mode and "privileged" mode
- In Unix:
  - When you login, you authenticate to the system by providing password
  - Once authenticated – create a shell for specific userID
  - All system calls pass userID to the kernel
  - Kernel checks and enforces authorization constraints
- Paradox
  - Any bug in the enforcer → you are hosed!
  - Make enforcer as small and simple as possible
    - Called the trusted computing base.
    - Easier to debug, but simple-minded protection (run a lot of services in privileged mode)
  - Support complex protection schemes
    - Hard to get it right!

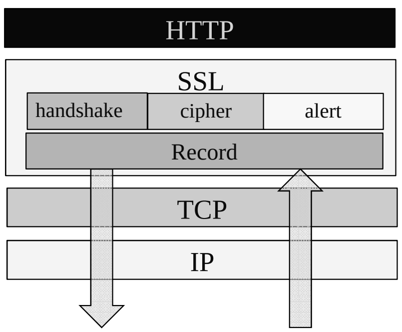
Dweeb Nolife develops a file system that responds to requests with digitally signed packets of data from a content provider. Any untrusted machine can serve the data and clients can verify that the packets they receive were signed. So utexas.edu can give signed copies of the read-only portions of its web site to untrusted servers. Dweeb's FS provides which property?

1. Authentication of file system users
2. Integrity of file system contents
3. Privacy of file system data & metadata
4. Authorization of access to data & metadata

Summary

- Security in distributed system is essential
- .. And is hard to achieve!

HTTPS (HTTP+SSL/TLS)



Client and Server encrypt traffic using Shared keys established by handshake protocol