Introduction, Syllabus and Course overview

- Instructor's information
  email: daw@cs
  Office: 629 Soda Hall
  Phone: 510-642-2758
  Course website: http://www.cs.berkeley.edu/~daw/teaching/cs261-f08/

- Grading
  Class Project: 40%
  Problem Sets: 35%
  Scribe Notes: 15%
  Paper summaries and class discussion: 10%

- Project:
  2-3 Person groups
  Anything reasonably related to class.
  Poster session and conference style presentation at semester's end.

- Problem Sets:
  2-4 Assignments throughout the semester.
  Turn in at beginning of class on due date. (No late assignments)

- Scribe Notes:
  Required to write scribe notes for one lecture.
  E-mail to professor within 1 week of lecture.

- Readings:
  No required textbook, all reading available through course website.
  Required to write summary for each assigned reading.
  Submit, on paper, at beginning of class that reading is due.

- Ethics

  - Feel free to interrupt during class in order to ask questions about class material.

Class Focus

- Primarily a systems perspective
  - Cryptography treated as “black-box.”

- Computer Security layers:
  Economics and Law relating to computer security
    ...
Secure Systems (CS 261)  
Cryptographic Protocols (A little CS 261, primarily CS 276)  
Cryptographic Primitives (CS 276)

- Additional Berkeley Courses
  Fall 2008:
  CS 294-22. Security, TU 2:00-5:00P, 320 Soda, Doug Tygar
  Web Security
  Spring 2009:
  CS 294. Network Security

- How to tell when you have a security problem?  
The presence of an adversary.

- Cryptography vs. Computer Security  
  Cryptography: Communication in the presence of an adversary.  
  Computer Security: Computing in the presence of an adversary.

- Security vs Reliability  
  Security: Protecting from an adversary.  
  Reliability: Typically, protecting from “mother nature.”

Goal
- Teach students how to build secure systems.  
- Understanding classic and new attacks.

Security Analysis
- How to do security analysis of a system (Security Evaluation).
  1. Understand the Security Goals  
     - What are you trying to achieve.  
     - What are you trying prevent? Ensure? Protect?
  2. Thread model  
     - Who
       Who is your adversary?  
       Who is your attacker?
     - What
       What types of attacks might they use?  
       What are their capabilities?  
       What are their limits?
     - Key: Determine what is in/out of scope.
  3. Did we achieve these goals?

Parts one and two can be categorized as “Requirements Analysis”  
while part three deals with a technical evaluation of a proposed
or deployed system.

- Example: Protection of a Bicycle.
  1. - Uptime
     - Availability for use
     - Replacement cost
     - Minimizing my expenses
     - Prevention and detection of tampering
     - Privacy

  2. Who:
     - Profession Bike Thief
     - Casual/Opportunistic thief
     - Mugger
     - Vandal
     - Police Officers

     Capabilities:
     - Access to tools
     - Access to the bicycles location
     - Access time to bicycle

     Limits:
     - What tools can they procure?
     - How noticeable is their attack?
     - Time required to execute attack
     - Economic viability of attack

     Motive:
     - Economics
       Make the cost to steal the bike greater then its value.
     - Enemy
     - Assassination

- Common Thread model categories
  Expertise
  Tools
  Motive
  Access: This could deal with physical or logical access as well as whether the attacker is considered an “Insider” or “Outsider” to an our organizational structure (ie: Company employee vs Unrelated Third party).

- The goal of the thread model is to determine what classes of threats do we want to defend against and which are out of scope.
3. 
   a. Reliance Analysis: At the architectural level, what components need to be relied on?
      - In the bicycle example this may be the lock as well as the bicycle frame, among other things.
   b. Look at the relied on components. (Scribe's Note: In fitting with 3a this might best be remembered at “Reliance Verification.”)

3 General Security Goals:
- Confidentiality
- Integrity
- Availability

Vocabulary
- Trust: Over-used. (Scribe's Note: Based on the professors common usage in class the best definition would be: A valuation of assured reliance.)
- Trustworthy: Equivalent to “Not going to fail.”
- Trusted: Equivalent to “Relied on.”
  Trusted components are often where the vulnerabilities lie.
  Even if you do not “trust” a component it may still be trusted due to system requirements.
  The general goal is to minimize the number of trusted components.

- Transitive Trust: “I trust Microsoft but do I trust everyone that Microsoft trusts?”
  If X trusts Y,
  and Y trusts Z,
  this does NOT imply X trusts Z.

Observations
- A common problem is legacy software. What was once trusted is no longer trustworthy.
- Often decisions are the result of the economic factors:
  What is our cost to defend?
  What is their cost to attack?