Evaluating Existing Systems

- Standards approaches aren't always suitable
- Not helpful for evaluating the security of running systems
- Not great for custom systems
- What do you do for those problems?

Two Different Kinds of Problems

1. I need to evaluate the design and implementation of the system
2. I need to evaluate what's going on in the system as it runs

Evaluating System Design Security

- Sometimes standards aren't the right choice
- What if you're building your own custom system?
- Or being paid to evaluate someone else's?
  - That's some companies' business
- This kind of review is about design and architecture
  - Evaluating running systems comes later

How Do You Evaluate a System's Security?

- Assuming you have high degree of access to a system
  - Because you built it or are working with those who did
- How and where do you start?
- Much of this material is from "The Art of Software Security Assessment," Dowd, McDonald, and Schuh

Stages of Review

- You can review a program's security at different stages in its life cycle
  - During design
  - Upon completion of the coding
  - When the program is in place and operational
- Different issues arise in each case

Design Reviews

- Done perhaps before there's any code
- Just a design
- Clearly won't discover coding bugs
- Clearly could discover fundamental flaws
- Also useful for prioritizing attention during later code review

Purpose of Design Review

- To identify security weaknesses in a planned software system
- Essentially, identifying threats to the system
- Performed by a process called *threat modeling*
- Usually (but not always) performed before system is built

Attack Surfaces

- Attackers have to get into your software somehow
- The more ways they can interact with the software, the more things you must protect
- Some entry points are more dangerous than others
  - E.g., those that lead to escalated privilege
- A combination of these factors defines a system's *attack surface*
- The smaller the attack surface, the better
  - But attack surface doesn't indicate actual flaws, just places where they could occur

Threat Modeling

- Done in various ways
- One way uses a five step process:
  1. Information collection
  2. Application architecture modeling
  3. Threat identification
  4. Documentation of findings
  5. Prioritizing the subsequent implementation review

1. Information Collection

- Collect all available information on design
- Try to identify:
  - Assets
  - Entry points
  - External entities
  - External trust levels
  - Major components
  - Use scenarios

One Approach[1]

- Draw an end-to-end deployment scenario
- Identify roles of those involved
- Identify key usage scenario
- Identify technologies to be used
- Identify application security mechanisms

[1]From http://msdn.microsoft.com/en-us/library/ms978527.aspx

Sources of Information

- Documentation
- Interviewing developers
- Standards documentation
- Source code profiling
  - If source already exists
- System profiling
  - If a working version is available

2. Application Architecture Modeling

- Using information gathered, develop understanding of the proposed architecture
- To identify design concerns
- And to prioritize later efforts
- Useful to document findings using some type of model

Modeling Tools for Design Review

- Markup languages (e.g., UML)
  - Particularly diagramming features
  - Used to describe OO classes and their interactions
  - Also components and uses
- Data flow diagrams
  - Used to describe where data goes and what happens to it

3. Threat Identification

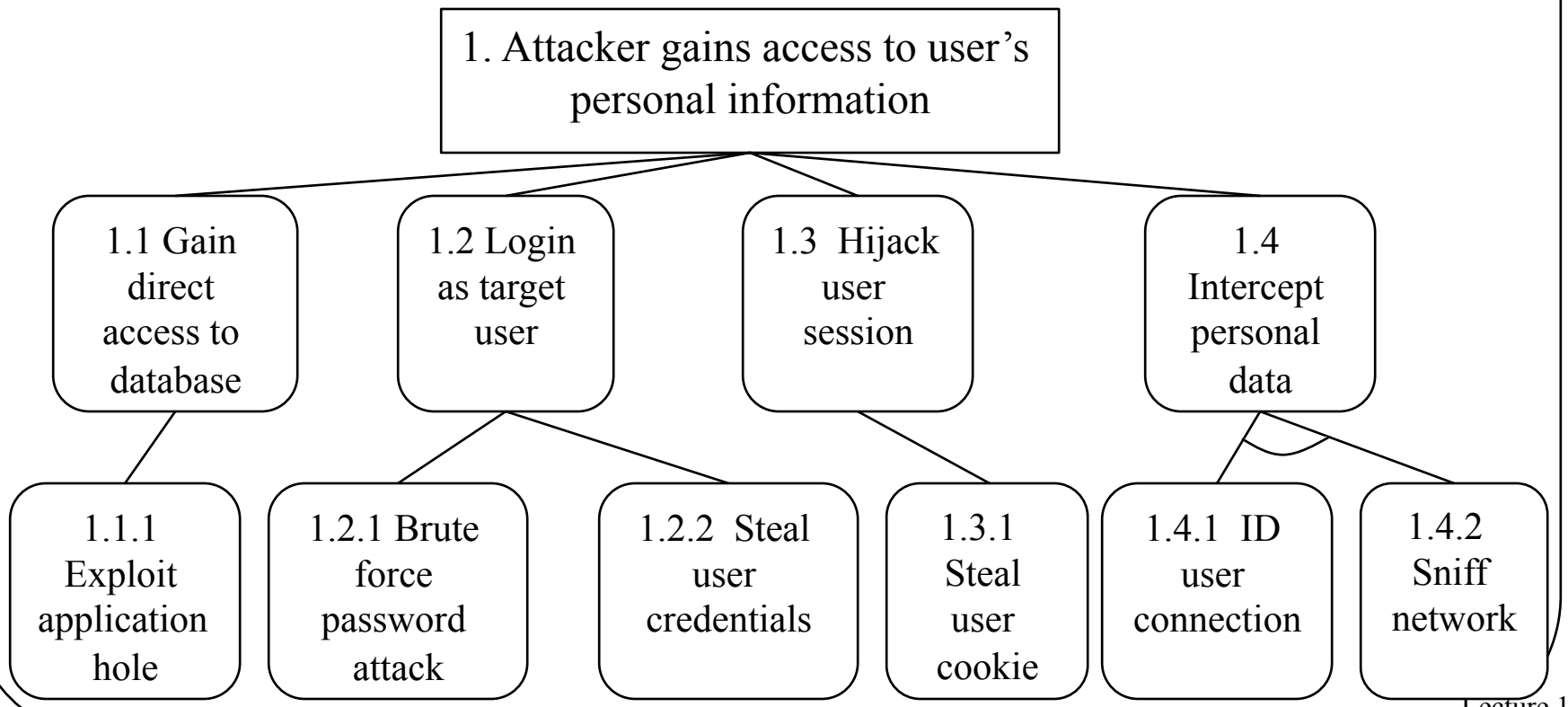
- Based on models and other information gathered
- Identify major security threats to the system's assets
- Sometimes done with *attack trees*

Attack Trees

- A way to codify and formalize possible attacks on a system
- Makes it easier to understand relative levels of threats
  - In terms of possible harm
  - And probability of occurring

A Sample Attack Tree

- For a web application involving a database
- Only one piece of the attack tree



The STRIDE Approach

- Developed and used by Microsoft
  - Part of their SDL threat modeling process[1]
- Depends on having built a good system model diagram
  - Showing components, data flows, interactions
  - Specifying where data and control cross trust boundaries
- Then, for each element, consider the STRIDE threats

[1]http://blogs.technet.com/b/security/archive/2012/08/23/microsoft-s-free-security-tools-threat-modeling.aspx

STRIDE Threats

- **S**poofing
- **T**ampering
- **R**epudiation
- **I**nformation Disclosure
- **D**enial of Service
- **E**scalation of Privilege

How To Apply STRIDE

- For each element in diagram, consider each possible STRIDE threat
- Some types of threats not applicable to some types of elements
- Pay particular attention to things happening across trust boundaries

4. Documentation of Findings

- Summarize threats found
  - Give recommendations on addressing each
- Generally best to prioritize threats
  - How do you determine priorities?
  - DREAD methodology is one way

DREAD Risk Ratings

- Assign number from 1-10 on these categories:
- **D**amage potential
- **R**eproducibility
- **E**xploitability
- **A**ffected users
- **D**iscoverability
- Then add the numbers up for an overall rating
- Gives better picture of important issues for each threat

5. Prioritizing Implementation Review

- Review of actual implementation once it's available
- Requires a lot of resources
- You probably can't look very closely at everything
- Need to decide where to focus limited amount of attention

One Prioritization Approach

- Make a list of the major components
- Identify which component each risk (identified earlier) belongs to
- Total the risk scores for categories
- Use the resulting numbers to prioritize