Lecture 2
September 23, 2022

Design Principles

- Simplicity, restriction
- Principles
  - Least Privilege
  - Fail-Safe Defaults
  - Economy of Mechanism
  - Complete Mediation
  - Open Design
  - Separation of Privilege
  - Least Common Mechanism
  - Least Astonishment

Overview

- Simplicity
  - Less to go wrong
  - Fewer possible inconsistencies
  - Easy to understand
- Restriction
  - Minimize access
  - Inhibit communication

Least Privilege

- A subject should be given only those privileges necessary to complete its task
  - Function, not identity, controls
  - Rights added as needed, discarded after use
  - Minimal protection domain

Related: Least Authority

- Principle of Least Authority (POLA)
  - Often considered the same as Principle of Least Privilege
  - Some make distinction:
    - *Permissions* control what subject can do to an object directly
    - *Authority* controls what influence a subject has over an object (directly or indirectly, through other subjects)

Fail-Safe Defaults

- Default action is to deny access
- If action fails, system as secure as when action began

Economy of Mechanism

- Keep it as simple as possible
  - KISS Principle
- Simpler means less can go wrong
  - And when errors occur, they are easier to understand and fix
- Interfaces and interactions

Complete Mediation

- Check every access
- Usually done once, on first action
  - UNIX: access checked on open, not checked thereafter
- If permissions change after, may get unauthorized access

Open Design

- Security should not depend on secrecy of design or implementation
  - Popularly misunderstood to mean that source code should be public
  - “Security through obscurity”
  - Does not apply to information such as passwords or cryptographic keys

Separation of Privilege

- Require multiple conditions to grant privilege
  - Separation of duty
  - Defense in depth

Least Common Mechanism

- Mechanisms should not be shared
  - Information can flow along shared channels
  - Covert channels
- Isolation
  - Virtual machines
  - Sandboxes

Least Astonishment

- Security mechanisms should be designed so users understand why the mechanism works the way it does, and using mechanism is simple
  - Hide complexity introduced by security mechanisms
  - Ease of installation, configuration, use
  - Human factors critical here

Related: Psychological Acceptability

- Security mechanisms should not add to difficulty of accessing resource
  - Idealistic, as most mechanisms add *some* difficulty
    - Even if only remembering a password
  - Principle of Least Astonishment accepts this
    - Asks whether the difficulty is unexpected or too much for relevant population of users

Key Points

- Principles of secure design underlie all security-related mechanisms
- Require:
  - Good understanding of goal of mechanism and environment in which it is to be used
  - Careful analysis and design
  - Careful implementation

Security Policy

- Policy partitions system states into:
  - Authorized (secure)
    - These are states the system can enter
  - Unauthorized (nonsecure)
    - If the system enters any of these states, it's a security violation
- Secure system
  - Starts in authorized state
  - Never enters unauthorized state

Confidentiality

- *X* set of entities, *I* information
- *I* has the *confidentiality* property with respect to *X* if no $x \in X$ can obtain information from *I*
- *I* can be disclosed to others
- Example:
  - *X* set of students
  - *I* final exam answer key
  - *I* is confidential with respect to *X* if students cannot obtain final exam answer key

Integrity

- *X* set of entities, *I* information
- *I* has the *integrity* property with respect to *X* if all $x \in X$ trust information in *I*
- Types of integrity:
  - Trust *I*, its conveyance and protection (data integrity)
  - *I* information about origin of something or an identity (origin integrity, authentication)
  - *I* resource: means resource functions as it should (assurance)

Availability

- *X* set of entities, *I* resource
- *I* has the *availability* property with respect to *X* if all $x \in X$ can access *I*
- Types of availability:
  - Traditional: *x* gets access or not
  - Quality of service: promised a level of access (for example, a specific level of bandwidth); *x* meets it or not, even though some access is achieved

Policy Models

- Abstract description of a policy or class of policies
- Focus on points of interest in policies
  - Security levels in multilevel security models
  - Separation of duty in Clark-Wilson model
  - Conflict of interest in Chinese Wall model

Mechanisms

- Entity or procedure that enforces some part of the security policy
  - Access controls (like bits to prevent someone from reading a homework file)
  - Disallowing people from bringing CDs and floppy disks into a computer facility to control what is placed on systems

Question

- Policy disallows cheating
  - Includes copying homework, with or without permission
- CS class has students do homework on computer
- Anne forgets to read-protect her homework file
- Bill copies it
- Who breached security?
  - Anne, Bill, or both?

Answer Part 1

- Bill clearly breached security
  - Policy forbids copying homework assignment
  - Bill did it
  - System entered unauthorized state (Bill having a copy of Anne's assignment)
- If not explicit in computer security policy, certainly implicit
  - Not credible that a unit of the university allows something that the university as a whole forbids, unless the unit explicitly says so

Answer Part 2

- Anne didn't protect her homework
  - Not required by security policy
- She didn't breach security
- If policy said students had to read-protect homework files, then Anne did breach security
  - She didn't do this

Types of Security Policies

- Military (governmental) security policy
  - Policy primarily protecting confidentiality
- Commercial security policy
  - Policy primarily protecting integrity
- Confidentiality policy
  - Policy protecting only confidentiality
- Integrity policy
  - Policy protecting only integrity