Chapter 12: Design Principles

- Overview
- Principles
  - Least Privilege
  - Fail-Safe Defaults
  - Economy of Mechanism
  - Complete Mediation
  - Open Design
  - Separation of Privilege
  - Least Common Mechanism
  - Psychological Acceptability

Acknowledgement: Matt Bishop

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
  - Example 1: capability amplification on Hydra
  - Example 2: setuid root applications on Linux

```
ls -l /usr/bin/passwd
rwsr-xr-x    root, root  47032 May 16   2017     /usr/bin/passwd
```

Fail-Safe Defaults

- Default action is to deny access
- If action fails, system as secure as when action began.
  - E.g., a mail server that fails to write to the spool directory should close the network connection and should NOT try to store the message elsewhere because this can be exploited by an attacker

Economy of Mechanism

- Keep it as simple as possible
  - KISS (Keep It Simple, Silly) Principle
- Simpler means less can go wrong
  - And when errors occur, they are easier to understand and fix
- Complex mechanisms are error-prone because they make incorrect assumptions about the environment:
  - Microsoft Applications: "about 10 - 20 defects per 1000 lines of code during in-house testing, and 0.5 defect per KLOC (KLOC IS CALLED AS 1000 lines of code) in released product (Moore 1992)."
  - e.g., allowing access based on the result of the *ident* protocol assumes that the originating host is trustworthy.
- Interfaces to and interactions with external entities can also cause problems

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
  - The opposite: “Security through obscurity”
  - Popularly misunderstood to mean that source code should be public
  - Does not apply to information such as passwords or cryptographic keys

Separation of Privilege

- Require multiple conditions to grant privilege
  - Separation of duty
    - Clark-Wilson model

Separation of Duty In CW Model

ER4 Only the certifier of a TP may change the list of entities (CDIs) associated with that TP. No certifier of a TP, or of an entity associated with that TP, may ever have execute permission with respect to that entity.

- Enforces separation of duty with respect to certified and allowed relations

Separation of Privilege

- Require multiple conditions to grant privilege
  - Separation of duty
    - Clark-Wilson model
  - Defense in depth, e.g., multi-factor authentication

Least Common Mechanism

- Mechanisms should not be shared
  - Reason: information can flow along shared channels
  - Covert channels: e.g., if two processes are not allowed to communicate but they share a resource, by coordinating access, they can communicate by modulating access to the entire resource.
  - Example: percent of CPU used. Two processes cannot talk directly, but they run on the same CPU. To send a 1 bit, the first process uses 75% of the CPU; to send a 0 bit, it uses 25% of the CPU. The other process sees how much of the CPU it can get and from that can tell what the first process used, and hence is sending
- Implementing this: isolation to reduce or eliminate sharing
  - Virtual machines
  - Sandboxes

Psychological Acceptability

- Security mechanisms should not add to difficulty of accessing resource
  - Hide complexity introduced by security mechanisms
  - Ease of installation, configuration, use
  - Human factors critical here
    - E.g., exceedingly long and complex passwords encourage users to write them on a piece of paper

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