Design Principles

ECS 153 Spring Quarter 2021

Module 2

Overview

- 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

Examples

- The UNIX/Linux user *root*: no access controls applied
- Mail server running as an ordinary user
  - May need to have *root* privileges to open port 25
  - Needs to be able to create files in spool directory

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

Example: Mail Spool Directory Full

- What to do
  - Notify client email is rejected due to full disk, and close connection
    - SMTP error code is 431
  - Notify administrator that spool directory cannot be written to as it is full
- What not to do
  - Increase privileges so it can store message elsewhere
  - Begin deleting old spooled mail messages

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

Examples

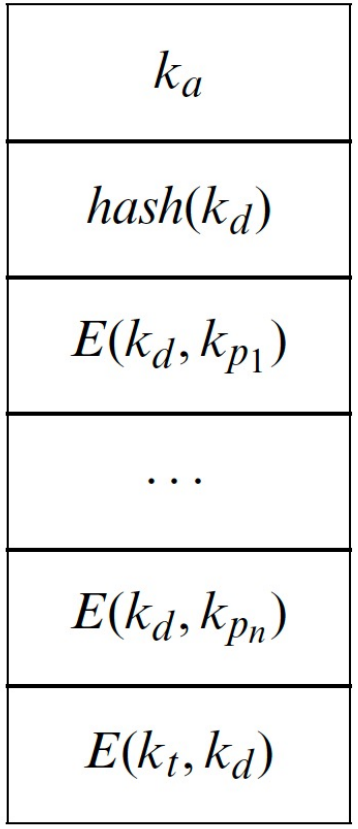
- When UNIX/Linux checks permissions to read, write a file
  - At open *only*
- DNS cache poisoning
  - Attacker inserts bogus DNS record in a reply
  - Victim contacts host with poisoned IP address
  - IP address is *not* revalidated so this goes to the wrong host

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
  - Plan for compromise of anything kept secret

Example

- DVD CSS
  - ka authentication key
  - kd disk key
  - E(kd, kpi) encrypted disk key for device
- Algorithm
  - Considered a trade secret
  - Norwegians derived compatible algorithm, made it freely available
  - Lawsuit filed in California court
    - Court posted filings on Internet, unless sealed
  - DVD CCA filed affidavit with actual algorithm
    - *and forgot to ask judge to seal it until a day later*



Separation of Privilege

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

Examples

- Company checks over $50,000 require 2 signatures
- FreeBSD: to become *root*, must meet 2 conditions
  - Know *root*'s password
  - Be a member of the *wheel* group (GID 0)

Least Common Mechanism

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

Examples

- Address Space Layout Randomization (ASLR)
  - Each instance of a program, when loaded in memory, has different addresses for functions
  - Attacker can’t use information about one process’ layout to attack another
- Site has only Windows 7 systems, all identical
  - So if attacker compromises 1, she can compromise all

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

Example

- Configuration file requires all times to be in minutes, except for one field that requires seconds
  - Actual instance: people often entered 0.5 (meaning 30 seconds) in the field
  - Program read the “0”, then stopped at the “.” as it ends an integer
  - Result: something that should have been flushed every 30 seconds was never flushed
- Hawai’i missile alert error

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