Lecture 4, April 6, 2026

Principles of Secure Design

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

Overview

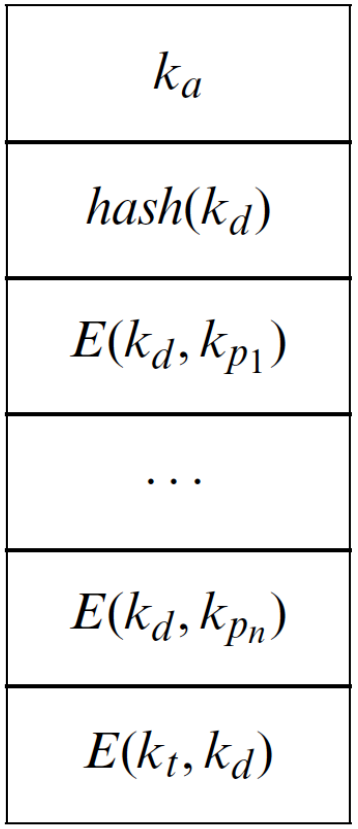
- Simplicity
  - Less to go wrong
  - Fewer possible inconsistencies
  - Easy to understand
- Restriction
  - Minimize access
  - Inhibit communication

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: Open Design

- 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: Separation of Privilege

- 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: Least Common Mechanism

- 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 11 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: Least Astonishment

- 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

What Is Privacy?

- Many different definitions; this seems to be the best one
- Privacy is the ability to control who has access to your personal information, *and what they can do with that information*

Privacy by Design

- Privacy cannot be added but must be designed into systems
- Privacy is more than just compliance with regulatory frameworks
- Objective for individuals: ensure strong privacy and control over one's information (Cavoukian, 2010)
- Objective for organizations: gain a sustainable, competitive advantage (Cavoukian, 2010)
  - By providing protection for customers', and their own, data
- Terminology: data subject is the person that the data is about or refers to

Principles

- Proactive not reactive; preventive not remedial
- Privacy as the default setting
- Privacy embedded into design
- Full functionality — positive-sum, not zero-sum
- End-to-end security — full lifecycle protection
- Visibility and transparency — keep it open
- Respect for user privacy — keep it user-centric

Proactive not Reactive

- Rather than reacting to privacy violations or attempted privacy violations, work to prevent them
- Requires identifying risks and developing countermeasures
  - The risks are to the data subjects as well as the companies
- The idea is to design and implement privacy controls to anticipate and block violations rather than compensate for them afterwards

Privacy as the Default

- Privacy controls should need to be loosened, not tightened
  - This is "opt in" rather than "opt out"
- So if someone does nothing, their data should be private
- Example: only web cookies necessary for functionality should be provided by default; cookies for advertising and sharing information with third parties should not be used unless the user indicates acceptance

Privacy Embedded into Design

- Like security, include privacy considerations in the requirements, and throughout the life cycle of development, implementation, distribution, maintenance, and retirement
  - The last is usually overlooked but *is critical*!
- Adding privacy after the steps does not work well
  - Privacy needs to be integrated into the system to ensure consistency and effectiveness

Full Functionality

- Think of privacy as functionality, not in conflict with functionality
  - "Zero sum" means you get privacy or you get functionality but not both
  - Better view: aim for both
  - Document any trade-offs, the reasons for the decisions made, and the expected effects of those decisions

End-to-End Security

- Privacy must be considered throughout the entire life cycle of the *data*
- Note the "life cycle" is that of the data and not of the system
  - Data may (and usually will) last beyond the system or software lifetime
- Data must be securely collected, saved, used, and when no longer needed, erased

Visibility and Transparency

- Everyone involved in the collection, storage, use, and erasing of the data must know what happens at each step
  - Particularly true of the uses to which the data may be put
  - This also includes data subjects (who the data is about)
- In particular, data subjects must know:
  - what data is being collected;
  - how the data will be used;
    - And ideally how it could be used!
  - how and where the data is to be stored; and
  - how the data will be disposed of when no longer needed.

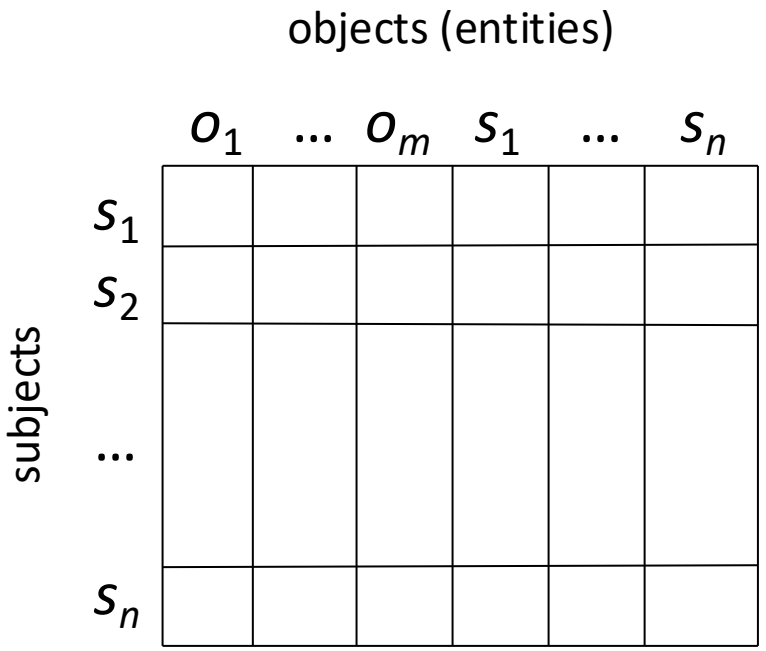
Respect for User Privacy

- Whenever possible, obtain user consent before collecting data
- Do not collect data unless it will be used
  - Minimizes risk, as compromise of data will only expose a minimum of private data
- Be accurate
  - Incorrect data can cause severe problems including false accusations, etc.
- Allow data subjects access to their data
- Make opting in or out simple and easy to understand

Access Control Matrix

- Abstraction that captures the protection state of the system
- Used for theoretical analyses
- Basis for access control mechanisms used in practice
  - More on this later

Description



- Subjects $S = \{ s_1, \ldots, s_n \}$
- Objects $O = \{ o_1, \ldots, o_m \}$
- Rights $R = \{ r_1, \ldots, r_k \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \ldots, r_y \}$ means subject s_i has rights $r_x, \ldots, r_y$ over object o_j

Example 1

- Processes *p*, *q*
- Files *f*, *g*
- Rights *r*, *w*, *x*, *a*, *o*

	f	*g*	*p*	*q*
p	*rwo*	*r*	*rwxo*	*w*
q	*a*	*ro*	*r*	*rwxo*

Example 2

- Host names *telegraph*, *nob*, *toadflax*
- Rights *own*, *ftp*, *nfs*, *mail*

	telegraph	*nob*	*toadflax*
telegraph	*own*	*ftp*	*ftp*
nob		*ftp, mail, nfs, own*	*ftp, nfs, mail*
toadflax		*ftp, mail*	*ftp, mail, nfs, own*

Example 3

- Procedures *inc_ctr*, *dec_ctr*, *manage*
- Variable *counter*
- Rights *+*, *–*, *call*

	counter	*inc_ctr*	*dec_ctr*	*manage*
inc_ctr	*+*			
dec_ctr	*–*			
manager		*call*	*call*	*call*

Boolean Expression Evaluation

- ACM controls access to database fields
  - Subjects have attributes
  - Verbs define type of access
  - Rules associated with objects, verb pair
- Subject attempts to access object
  - Rule for object, verb evaluated, grants or denies access

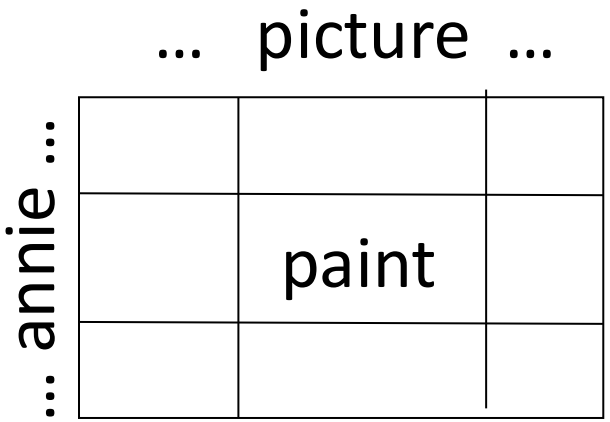
Example

- Subject annie
  - Attributes *role* (artist), *group* (creative)
- Verb paint
  - Default 0 (deny unless explicitly granted)
- Object picture
  - Rule:

    paint: ‘artist’ in subject.role and
    ‘creative’ in subject.groups and
    time.hour ≥ 0 and time.hour ≤ 4

ACM at 3AM and 10AM

At 3AM, time condition met
ACM is:



At 10AM, time condition not met
ACM is: