# Security

# Policies and Mechanisms

- Policy says what is, and is not, allowed
  - This defines “security” for the site/system/*etc*.
- Mechanisms enforce policies
- Composition of policies
  - If policies conflict, discrepancies may create security vulnerabilities

# Goals of Security

- Prevention
  - Prevent attackers from violating security policy
- Detection
  - Detect attackers violating security policy
- Recovery
  - Stop attack, assess and repair damage
  - Continue to function correctly even if attack succeeds

# Assumptions and Trust

- Underlie *all* aspects of security
- Policies
  - Unambiguously partition system states
  - Correctly capture security requirements
- Mechanisms
  - Assumed to enforce policy
  - Support mechanisms work correctly

# Requirements

- Trusted Computer Security Evaluation Criteria (TCSEC)
  - And its derivatives, the “Rainbow Series”
- FIPS 140
  - For cryptographic implementations
- Common Criteria
  - For systems that match protection profiles
- System Security Engineering Capability Maturity Model (SSE-CMM)
  - For processes used to develop systems
- GDPR and CCPA
  - Laws in the EU and California that govern privacy

# Design Principles

- Least privilege
  - Process should be given only those privileges necessary to complete its task
- Fail-safe defaults
  - Default is to deny permission
  - If action fails, system stays as secure as when action began
- Economy of mechanism
  - Keep things as simple as possible (KISS principle)
- Complete mediation
  - Check permissions on every access

# Design Principles

- Open design
  - Security should not depend on secrecy of design or implementation
- Separation of privilege
  - Require multiple conditions to hold in order to grant privilege
- Least common mechanism
  - Minimize sharing of resources
- Least astonishment
  - Security mechanisms should be designed so users understand why the mechanism works the way it does, and using mechanism is simple
  - Earlier version: principle of psychological acceptability, which says security mechanisms should not add to difficulty of accessing resource

# User or Subject Authentication

- Authentication: binding of identity to subject
  - Identity is that of external entity (my identity, Matt, *etc*.)
  - Subject is computer entity (process, *etc*.)

# Establishing Identity

- One or more of the following
  - What entity knows (*eg.* password)
  - What entity has (*eg.* badge, smart card)
  - What entity is (*eg.* fingerprints, retinal characteristics)
  - Where entity is (*eg*. In front of a particular terminal)

# Passwords

- Sequence of characters
  - Examples: 10 digits, a string of letters, *etc*.
  - Generated randomly, by user, by computer with user input
- Sequence of words
  - Examples: pass-phrases
- Algorithms
  - Examples: challenge-response, one-time passwords

# Storage

- Store as cleartext
  - If password file compromised, *all* passwords revealed
- Encipher file
  - Need to have decipherment, encipherment keys in memory
  - Reduces to previous problem
- Store one-way hash of password
  - If file read, attacker must still guess passwords or invert the hash

# Approaches: Password Selection

- Random selection
  - Any password from *A* equally likely to be selected
- Pronounceable passwords
- User selection of passwords

# Random Passwords

- Choose characters randomly from a set of possible characters; may also choose length randomly from a set of possible lengths
- Expected time to guess password maximized when selection of characters in the set, lengths in the set, are equiprobable
- In practice, several factors to be considered:
  - If password too short, likely to be guessed
  - Some other classes of passwords need to be eliminated, such as repeated patterns (“aaaaa”), known patterns (“qwerty”)
  - But if too much is excluded, space of possible passwords becomes small enough to search exhaustively