Information Securit I Notes

Gregory Ecklund

October 16, 2024

1 Fundamental Security Design Principles

- Economy of mechanism
  - Simple and as small as possible
  - Complex designs increase cost, time, testing needs and the likelihood of an exploit or bug
- Fail-safe defaults
  - Access decisions should be based on permission rather than exclusion
  - The default is lack of access with permissions added
  - This approach exhibits better fail-safe characteristics
- Complete mediation
  - Every access must be checked against access control mechanism
  - Access permissions should not be derived from a cache
  - If the system depends on cached information it changes in authority will be less immediate
- Open design
  - An open design has a higher degree of professional scrutiny
  - Standardization also promotes system compatibility
  - Tried and tested is better than bespoke and unknown
- Separation of privilege
  - Multiple privilege attributes are required to access a restricted resource
  - Also, high privilege operations in separate processes with a higher level of privilege needed
    - User
    - Super User
    - Administrator

- Least privilege
  - Every process and user should operate using the least set of privileges to perform the task
  - Roles and associated privileges should be defined in security policies
  - Administrators with special privileges should be granted those 'special' privileges only when needed
    * Leaving special privileges on all the time increases risk
- Least common mechanism
  - Minimize the amount of mechanism common to more than one user and depended on by all users
  - A shared mechanism is a potential information path between users and should be designed with care
  - For example, the same authentication method should not be used to authenticate employers of a company and non-employee users of its popular website
- Psychological acceptability
  - Security methods must
    * Be user friendly
    * Not interfere with work of users
    * Meet the needs of those who authorize access
- Isolation
  - Access systems should be isolated (physically and logically) from critical resources to prevent disclosure or harm caused by tampering
  - Processes and files of users should be isolated from each other
- Encapsulation
  - A basic form of isolation rooted in object oriented programming practices

    - Keeping class and object members private and hidden unless there is a reason not to
    - Use setter methods (mutators) to ensure fields set within valid, safe values
    - Each class having one well defined purpose
  - All of this prevents accidental or miss intended use of an application and its functions

- Modularity
  - Security functions should embody a modular architecture for mechanism implementation
  - Common security modules such as cryptographic algorithms developed for reuse
  - Coding to an interface allows easy substitution for other implementations
  - Code for reuse. Less code repetition leads to a safer system, that is less prone to error and is easier to test and maintain

- Layering
  - Layering refers to multiple overlapping security protections
  - The failure or circumnavigation of one system will not leave the rest unprotected
  - **Layers:**
    1. Physical Security
    2. Identity and Access
    3. Perimeter
    4. Network
    5. Compute
    6. Application
    7. Data

- Least astonishment
  - A program or user interface should always respond in a way least likely to astonish the user

- For example, it should be transparent enough for the user to see how the security goals map to the provided security mechanism
- This principle can also be applied to code modules used by other developers