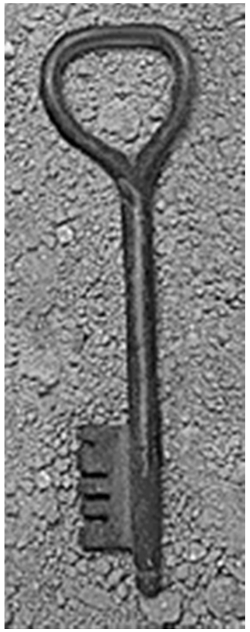
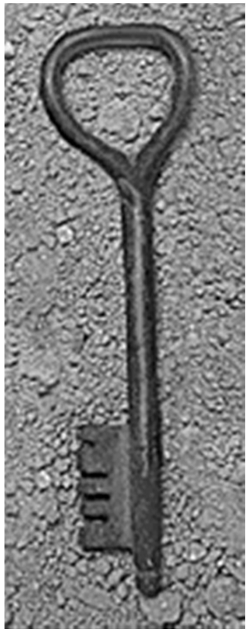
Trusted Operating Systems



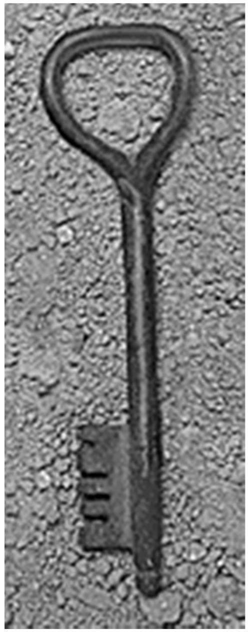
Agenda

- Trust vs. security
- Designing trusted systems.



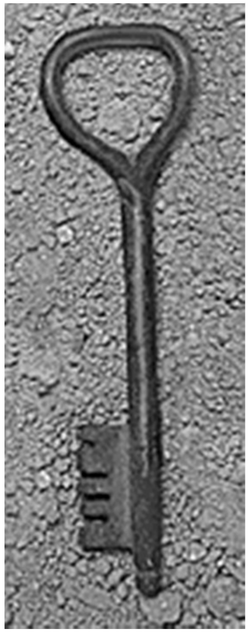
Trust vs. Security

- Trust is **relative**: *I trust him more than him …*
- Trust is **specific**: *I trust her to close the door …*
- Trust is a *characteristic* of a system, whereas security is a *goal.*
- So, people tend to prefer to call operating systems with particular security features *trusted.*
- For example. trust can mean something quite different in *military* and *commercial* contexts.
  - Remember MAC/DAC ?



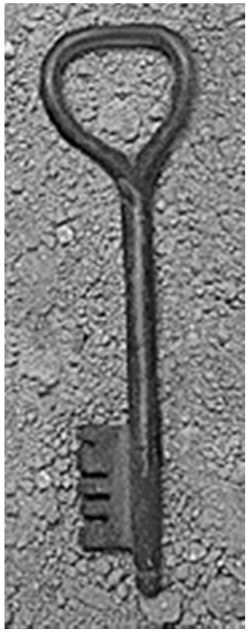
Trusted Operating Systems

- Operating systems not defined as “secure”.
- The term “trusted” is used.
- The issue is whether the system can be trusted by the users to provide the required level of security.



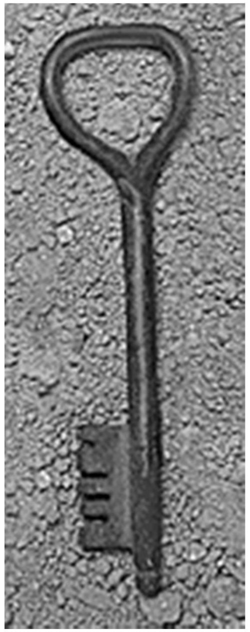
Agenda

- ♦ Trust vs. security
- ♦ Designing trusted systems.



Design Principles

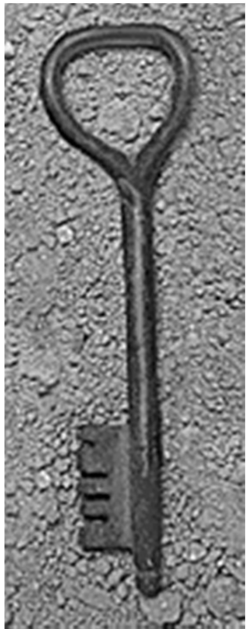
- Overview
- Principles
  - Economy of Mechanism
  - Fail-Safe Defaults
  - Complete Mediation
  - Open Design
  - Separation of Privilege
  - Least Privilege
  - Least Common Mechanism
  - Psychological Acceptability



Overview

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

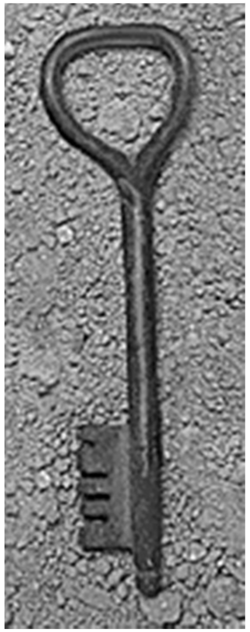
Saltzer and Schroeder 1975



1. Economy of Mechanism

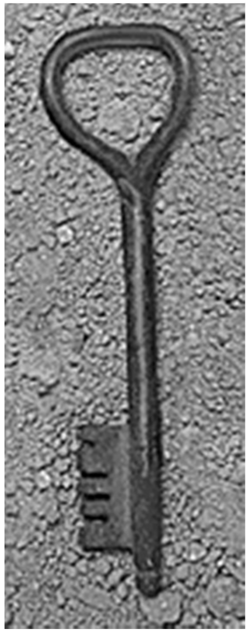
- *Keep the design as simple and small as possible*
- Simpler means less can go wrong
  - And when errors occur, they are easier to understand and fix
- Interfaces and interactions

KISS principle (Keep it Simple, Silly)



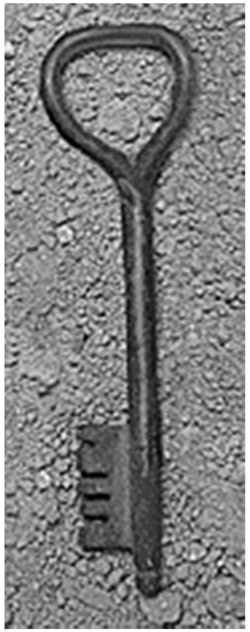
Example: Voting Software

- Diebold Accuvote TS
  - 31,000 lines of C++
- PRUI (Yee *et al*)
  - < 300 lines of Python
- Which one would you trust? (Att: I’m not talking about which one is secure.)



2. Fail-Safe Defaults

- *Base access decisions on permission rather than exclusion*
- Burden of proof is on the principal seeking permission
- If the protection system fails, then legitimate access is denied but illegitimate access is also denied



Examples

- Remove illegal characters:

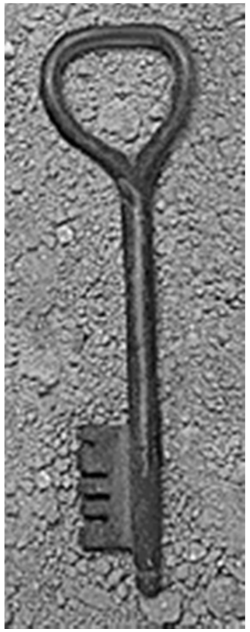
  illegal_chars = “,;/\\!”

  str = [c from input if c not in illegal_chars]

- Better:

  legal_chars = “abcdefg…”

  str = [c from input if c in legal_chars]



More Examples - User Data in SQL Queries

- set UserFound=execute(

  SELECT * FROM UserTable WHERE

  username=′ ″ & form(“user″) & “ ′ AND

  password=′ ″ & form(“pwd″) & “ ′ ″);

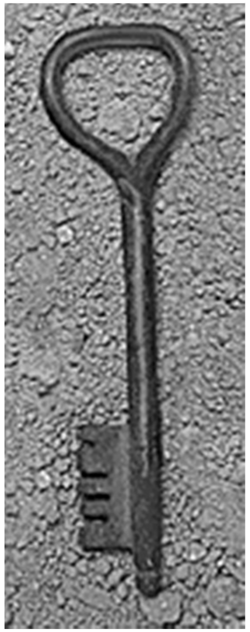
  – User supplies username and password, this SQL query checks if user/password combination is in the database

- If not UserFound.EOF

  Authentication correct

  else Fail

Only true if the result of SQL query is not empty, i.e., user/pwd is in the database



More Examples - SQL Injection

Always true!

- User gives username ′ OR 1=1 --
- Web server executes query

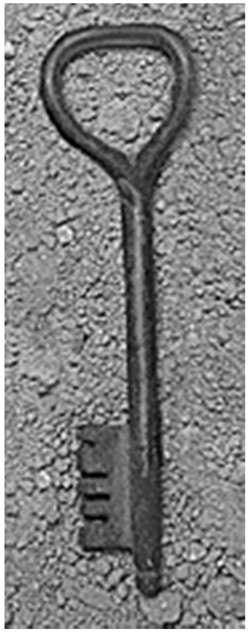
set UserFound=execute(

SELECT * FROM UserTable WHERE

username=′ ′ OR 1=1 -- ...);

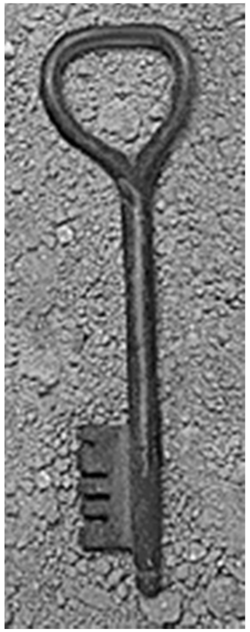
Everything after -- is ignored!

- This returns the entire database!
- UserFound.EOF is always false; authentication is always “correct”



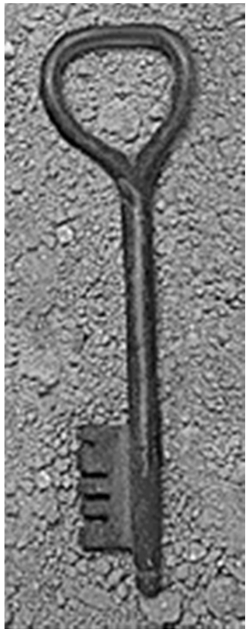
3. Complete Mediation

- *Every access to every object must be checked for authority*
- Usually done once, on first action
  - UNIX: access checked on open, not checked thereafter
- If permissions change after, may get unauthorized access
- Proposals to gain performance by remembering the result of an authority check should be examined skeptically



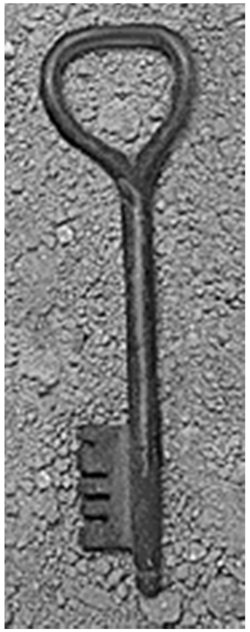
4. Open Design

- *The design should not be secret*
  - Design / code should be available for public review
  - Easier to achieve assurance
- *The mechanisms should not depend on the ignorance of potential attackers, but rather on the possession of specific, more easily protected, keys or passwords.*
  - Kerckhoffs' principle: do not depend on the secrecy of something you cannot change



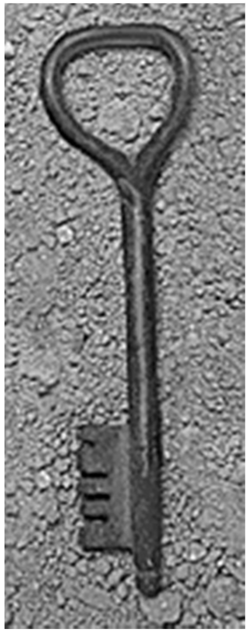
Examples

- GSM: Algorithms designed in secret
  - A3/A8: reverse engineered (‘98), broken 3 hours later
  - A5/2: reverse engineered (‘99), broken 5 hours later
  - A5/1: reverse engineered (‘99), broken 1 year later
- CSS algorithm in DVD



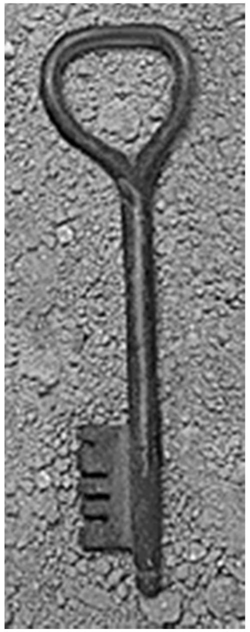
5. Separation of Privilege

- *Where feasible, a protection mechanism that requires two keys to unlock it is more robust and flexible than one that allows access to the presenter of only a single key.*
- Require multiple conditions to grant privilege
  - Separation of duty
  - Defence in depth



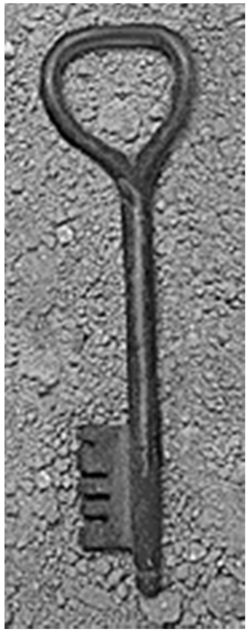
Examples

- root / admin account
  - Most operating systems use *admin* account
  - Any privileged action requires admin privileges
    - All-or-nothing access
- Check must be signed by 2 officers if amount > 75K $
- Two or more keys for the gate of cashbox



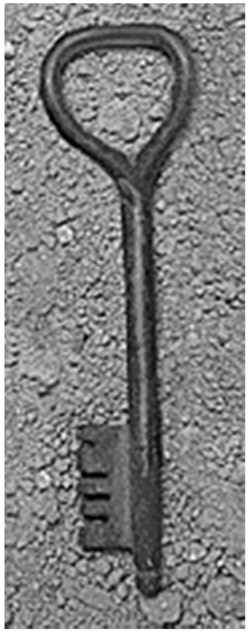
6. Least Privilege

- *Every program and every user of the system should operate using the least set of privileges necessary to complete the job*
- 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



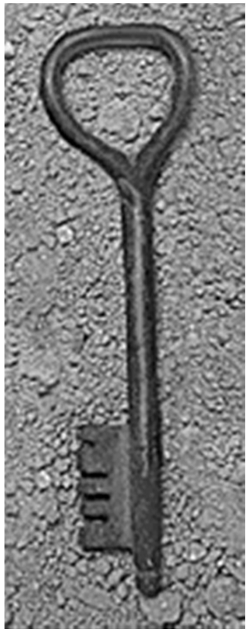
Examples

- Military: need-to-know
  - BLP/Biba compartments
- PitBull: proxy privilege
  - A new process defines subset of parent's privilege that it needs to use
- Others?



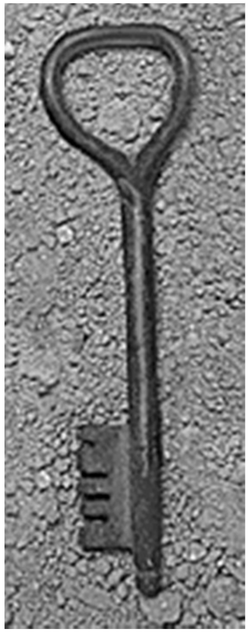
7. Least Common Mechanism

- Minimize the amount of mechanism common to more than one user and depended on by all users
  - Every shared mechanism (especially one involving shared variables) represents a potential information path between users (remember covert channel?)
  - Further, any mechanism serving all users must be certified to the satisfaction of every user, a job presumably harder than satisfying only one or a few users.



8. Psychological Acceptability

- *It is essential that the human interface be designed for ease of use so that users routinely and automatically accept the protection mechanisms correctly*
- 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



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