Security Principles

Security is a system requirement just like performance, capability, cost, etc.

Therefore, it may be necessary to trade off certain security requirements to gain others

Design Principles for Protection Mechanisms

• Least privilege
• Economy of mechanism
• Complete mediation
• Open design
• Separation of privilege
• Least common mechanism
• Psychological acceptability
• Fail-safe defaults

Saltzer and Schroeder 1975

Least Privilege

• Should only have the rights necessary to complete your task.
• Default should be lack of access

• If access needed temporarily, then it should be rescinded right after use

Economy of Mechanism

• Sufficiently small and simple as to be verified and implemented
  – e.g., security kernel
• Simpler means less can go wrong
  – And when errors occur, they are easier to understand and fix

Economy of Mechanism (2)

• Complex mechanisms may not be correctly:
  – Understood
  – Modeled
  – Configured
  – Implemented
  – Used
• Keep it as simple as possible
  – KISS Principle

Complete Mediation

• Every access to every object must be checked
• Must be efficient
• In addition to normal runtime, must be done at:
  – initialization
  – shutdown
  – restart
Open Design

• Don’t depend on secrecy of the design
• “Security through obscurity” is a bad idea
• Should be open for scrutiny by the community
• Better to have a friend/colleague find an error than a foe

Diebold Voting Machines

Inspection of the code by John Hopkins team found
– Passwords embedded in the source code
– Unauthorized privilege escalation and other vulnerabilities
– Incorrect use of cryptography
– Undetected, unlimited votes by voters
– Insider threats - company workers or election officials can alter voters’ ballot choices without their knowledge

Separation of Privilege

• Access to objects should depend on more than one condition being satisfied
  – Separation of duty
  – Two person rule

Least Common Mechanism

• Minimize the amount of mechanism common to more than one user and depended on by all users
• Every shared mechanism is a potential information path

Psychological Acceptability

• User interface must be easy to use, so that users routinely and automatically apply the mechanisms correctly. Otherwise, they will be bypassed
• Security mechanisms should not add to difficulty of accessing resource

Fail-Safe Defaults

• The default is lack of access
• Need to argue why a user should have access. Do not argue why a user should not have access
• If action fails, system as secure as when action began
**Principles for a Secure Design**

- Design security in from the start
- Allow for future security enhancements
- Minimize and isolate security controls
- Employ least privilege
- Structure the security relevant features
- Make security friendly
- Don’t depend on secrecy for security  

*Morrie Gasser 1988*

---

**To Make Security Friendly**

- Security should not impact users who obey the rules
- It should be easy for users to give access
- It should be easy for users to restrict access
- Established defaults should be reasonable

---

**Principles for Software Security**

- Secure the weakest link
- Practice defense in depth
- Fail securely
- Follow the principle of least privilege
- Compartmentalize
- Keep it simple
- Promote privacy
- Remember that hiding secrets is hard
- Be reluctant to trust
- Use your community resources  

*Viega and McGraw 2001*

---

**Secure the Weakest Link**

- A software security system is only as strong as its weakest link
- Attackers go after the easy targets
  - they will go after endpoints rather than trying to crack encryption
  - they will attempt to crack an application visible through the firewall rather than firewall itself
- Identify and strengthen weak links until an acceptable level of risk is achieved

---

**Practice Defense in Depth**

- Use diverse defensive strategies
- If one layer turns out to be inadequate, another layer will hopefully prevent a complete compromise
  - firewall to protect subnet, but sensitive information on the subnet is encrypted
- DoD Eligible Receiver experience 1997

---

**Fail Securely**

- If your software has to fail, make sure it does it securely
Follow the Principle of Least Privilege

- Only the minimum amount of access necessary to perform an operation should be granted, and that access should be granted only for the minimum amount of time necessary.

Compartmentalize

- Basic access building block is not all or nothing.
- Minimize the amount of damage that can be done by breaking the system into units.
- Very few operating systems do this because it is difficult to manage.
- Root privilege is an example of how not to do it.

Keep it Simple

- Complex design is never easy to understand.

Promote Privacy

- Try not to do anything that compromises the privacy of the user.
- Often trades off against usability:
  - System should forget credit card numbers.
  - Users hate having to type it in each time.
- Should extend to systems and code:
  - No reason to give out any more information than necessary (e.g., os name and version).

Remember that Hiding Secrets is Hard

- Skilled youth can circumvent any protection that a company tries to hardcode into their software (e.g., DVD viewers).
- Binaries can be reverse engineered.

Be Reluctant to Trust

- Instead of making assumptions that need to hold true, you should be reluctant to extend trust.
- How can any COTS component be trusted to be secure?
- Just because a particular security feature is an emerging standard doesn’t mean it actually makes sense.
- Sometimes it is prudent not to trust yourself.
Use Your Community Resources

- Repeated use without failure promotes trust
- Public scrutiny promotes trust