



TEKNILLINEN KORKEAKOULU

---

# Safe systems



# Design Principles

---

- From "Computer Security, Art and Science" by Matt Bishop
- And from "The Protection of Information in Computer Systems" by J. Saltzer and M. Schroeder
- System design should be guided by simplicity and restriction
  - A simple system has less room for things to go wrong



# Design Principles 1-2

---

- 1. Principle of Least Privilege
  - A subject should be given only those privileges that it needs in order to complete its task
    - E.g. service processes should be run under separate user-ids, which have limited access to the rest of the system
- 2. Principle of Fail-Safe Defaults
  - Unless a subject is given explicit access to an object, it should be denied access to that object
    - The design should err on the conservative side
    - Also when an application fails to reach its objective, it should leave the system in a safe state



# Design Principles 3-5

---

- 3. Principle of Economy of Mechanism
  - Security mechanisms should be as simple as possible
    - Less complexity means less openings for an attacker
    - Less code to verify or walk through
- 4. Principle of Complete Mediation
  - All access to objects should be checked to ensure that they are allowed
    - The checking should be done for each transaction, not just in the startup phase of an application
    - Beware of cached decisions
- 5. Principle of Open Design
  - The security of a mechanism should not depend on the secrecy of its design or implementation
    - E.g. DVD encryption



# Design Principles 6-8

---

- 6. Principle of Separation of Privilege
  - A system should not grant permission based on a single condition
    - E.g. on some Unix systems to change from user to root requires both the password and that the user belongs to a specific group
    - Equal to "two key" systems used for nuclear weapons to minimize accidental actions
- 7. Principle of Least Common Mechanism
  - Mechanisms used to access resources should not be shared
    - E.g. instead having all services use the same operating system, virtual machines can be used
- 8. Principle of Psychological Acceptability
  - Security mechanisms should not make the resource more difficult to access than if the security mechanisms were not present



- Least Privilege
  - Fail-Safe Defaults
  - Economy of Mechanism
  - Complete Mediation
  - Open Design
  - Separation of Privilege
  - Least Common Mechanism
  - Psychological Acceptability
- How would you apply these when designing a WWW based service?



# Why are not all systems designed according to these design principles?

---

- Design effort, "time to market" dictates that development effort must be focused
  - Security has not been considered an important issue in the design of many systems that are currently popular
- Usability issues
  - Security features are often seen as counterproductive
  - The complete access provided to administrators in some systems makes also their work much more easier
- Performance
  - Security checks in code require both design and CPU capacity



# What is a System?

---

- For this presentation, a system is the hardware, software and procedures that are used to process information
  - The system can easily consist of hundreds of people and computers with connecting networks, procedures for accepting and permitting things to happen etc.
- The data is manipulated in different parts of the system
- All points of the system are susceptible to attack



# When to Attack a System

---

- During the
  - Design
  - Implementation & installation
  - Use
  - Afterwards



# How to Attack the System Design

---

- The early bird catches the worm
- It is easier to leave things out of the design than to try to include new features
  - The IPSec encryption standard requires only DES and “none” algorithms to be implemented in all products
  - Early smart card designs did not address the issue of measuring the smart card power usage. The keys could be deduced using the Chinese Remainder Theorem.
- Influencing the design requires long term planning and the ability to use the weakened design at some later point
  - Influence might come as well from inside as outside
  - National governments and large corporations
- A common defense is independent or peer design review



# Implementation and Installation

---

- Designs do not usually cover all possible implementation and installation issues
  - A program's specification might leave room in the source code for a variable that is set when a certain sequence of events happens and that is later used to authorize a transaction
- Authorization might be exceeded
  - The classic "embassy mix" of 90% concrete and 10% microphones
- Peer and independent review are used here too
- Good design makes misuse more difficult
- Traditionally this phase is often the focus of security efforts
  - The effort might be better spent in other parts of the system life



# How to Attack the System in use

---

- The most common target to those attackers, who did not start planning early enough
- Largest problem is trusted people, insiders
- There is a huge amount of possible attacks
  - Have authorized persons make unauthorized actions
  - Gather information using the side-band channels of the system
  - Enter false data to the system
- Prevention comes from having well thought out procedures and from following those procedures
  - People are the issue here



# How to Attack the System Afterwards

---

- Old data has often value
  - Public persons' medical records
  - Military plans
- The design of a system should include also how to dispose the system components
  - This doesn't usually happen
- Also many commercial systems attempt to reach customer lock in, by making it difficult or impossible to extract data
  - Attention should be paid to this during the design and product selection phase of a system



- After the initial design phase a system is often re-designed to meet new criteria
- Large systems are often in a permanent state of flux, with new features and requirements being designed and implemented concurrently with production use and maintenance
- Security managers need to understand that the system is never in sync with the specifications
  - Managing the security in the transition phase is a challenge
  - Often it is possible to walk in to a secure site during the construction process



# Example of Change Management

---

- Modern service production environments implement separation of privilege in the process to protect the system from attacks during changes
  - New software is developed in a separate environment
  - After development the binary code is copied to a testing environment
  - After testing the code is copied to the production environment
    - Not copied over old, rollback must be possible
  - Any employee may have access to only one of these systems



# Requirements and solutions

- Could we solve everything like this?

<b>We require</b>	<b>We use</b>
Confidentiality	Cryptography
Integrity	Firewalls and checksums
Availability	Backups
Accountability	User identification and logs

No!



- An e-commerce system requires protecting credit card numbers
  - Strong crypto can protect data confidentiality and integrity in transit
  - PKI provides authentication services
- Is an e-commerce system that uses PKI to authenticate both the customer and the server and cryptography to protect the credit card number secure?
- No, because the attacker can target the store's computer systems itself
- Solving part of the problem is not enough!



# Comprehensive solutions

---

- The security solution must cover all possible approaches to be useful
- In ordinary systems design the question is "what must happen for this system to work"
- In security design the question is "what must not happen so that we can consider this system to be working"
  - Security design should be considered ordinary system design



# Security Assurance

---

- Security assurance is a process that makes an effort to verify the security of a system at a reasonable level
- A proof of the security of the complete design is impossible in the real world
  - Any safe can be cracked with large enough hammer
  - Future inventions can make old solutions obsolete
- Some starting points for assurance [Anderson, Security Engineering ch. 23.2.1]
  - Functionality
  - Strength of mechanisms
  - Implementation
  - Usability



# Usability in Security Design

---

- Does changing passwords every month make them more secure?
- The SET standard was designed to be secure, is anybody using it?
- A decrease in the usability level of a system is part of the cost of the security system, but the cost must be according to the risks



# Assurance is a continuous Process

---

- All parts of the lifecycle must be covered
- All parts of the system must be covered
  - Software network interfaces
  - User training
  - Backup handling
  - Physical access
  - etc.
- There are security standards
  - Provide "Best Practice" information, checklists, measurements
- The standards are useful if used correctly, but they do not really prove quality of a product



# Systems platforms

---

- The system design is implemented on a platform
  - Hardware
  - Operating systems
  - Software components
- The existing and missing security features of these components must be included in the design



# Hardware security

---

- Usually relevant when untrusted people have access to the physical components of the system
- A web server in a locked and guarded environment does not usually need special hardware features
- A telephone SIM card has financial importance and is likely to be a target for misuse
  - The solution is
    - Make the SIM card tamper resistant raising the cost of misuse
    - Install fraud detection capability to the telephone network, limiting the exposure to risk



# Software components

---

- Most operating systems have been designed before widespread networks existed
  - The basic protection is not very good
  - Most common way to break the security is the buffer overflow attack
- Some component systems, like ActiveX are basically flawed and do not survive in a networked environment
- There is no easy way to know which components are good
  - Secure systems are built by people who have experience in building secure systems



# Basic Operating System Flaws

---

- Unix and Windows families do not provide a strong internal security architecture
  - In both operating systems there is too much programming code running in the super user or unrestricted mode (root, kernel)
  - They do not support multilevel security and multilateral security support is weak
- Operating systems provide too many services
  - Any service is an potential exploit that can be used to gain unauthorized access
  - Both internal and network services



# Solving the OS Flaws

---

- Remove all unnecessary functionality (usually cheap)
- Separate different security levels to separate physical computers and limit communications (more expensive)
  - Front end web servers handle user requests and transmit actual data to back end databases
  - Separate logging and backup servers
- Use special operating systems (expensive)
  - Less support and software components, more difficult to find trained people



- Many security problems are caused by bad-quality software
- 95 % of everything is junk, 5 % is good
  - Some optimists quote 80 / 20
  - Some pessimists quote 99 / 1
- There are quite effective methods against this
  - Normal quality assurance procedures
  - Good design methods
  - Peer review
  - Auditing



- Requirement: confidentiality of a credit card transaction in an e-commerce system
- Threat: outside attacker gains access to the number of the credit card by:
  - Eavesdropping to the network traffic
  - Breaking the SSL protection
  - Breaking to the commerce server
  - Retrieving papers from the trash
  - Stealing the customer's wallet
  - Asking the customer for the credit card number
  - etc.
- Risk management: Which consequences can we accept



- Eavesdropping:
  - Can be solved by providing confidentiality, integrity and server side authentication to the connection
  - for example SSL, SSH, VPN solutions
- Breaking the SSL crypto
  - Assumed to be more expensive than the potential benefit of the credit card number to the attacker
- Breaking to the commerce server
  - Network and service design limits access to the server
  - Credit card numbers are not stored in the front end server
  - The network connection between the front and back end servers is monitored by an intrusion detection system
  - Operators are trained properly



- Retrieving papers from trash
  - Proper information handling procedures
- Stealing the customer's wallet
  - Not the e-commerce server's problem
- Asking the customer for the number
  - If the attacker can masquerade as the e-commerce server, this is a real problem (bad publicity affects revenues)
  - Training the users to use SSL is one solution
  - Using SET could be a solution



- Other design issues
  - How to identify that the customer has not stolen a credit card
  - Availability: backups, redundancy
  - Responsibilities: transaction and other logging
  - Privacy issues
  - Did you notice that the requirement was to protect the confidentiality of the transaction, not just the credit card number
  - And plenty more issues



# Sources of Security information

---

- CERT (recommended!), <http://www.cert.org/>
  - Publishes information on threats when a solution is available
- CERT-FI, <http://www.ficora.fi/>
  - CERT for Finland, by the ministry of communications
- Network discussion forums
  - Bugtraq, NT Bugtraq, Securityfocus and many others
  - Usually contain much noise, quality varies
  - Up to date information about problems even before solutions
- Courses, seminars, workshops, organizations
  - <http://www.sans.org/>
  - <http://www.usenix.org/>
  - Also professional qualification courses



# ...Sources of Security information

---

- Books
  - Trick is to find the worthwhile ones
- Standards
  - BS-7799 (ISO 2700[012]), SSE-CMM, Common Criteria, ITSEC
- Security consultants
  - Focus on timely information
- Vendors
  - Patches, updates, virus information
- The organization should have a strategy for collecting the up to date information in timely manner



- One hole is all it takes for the attacker to crack the system
  - You need to find and fix all bugs and holes
- Evaluate your needs and pick known, good solutions and configurations
- Keep it simple