Secure Systems

Idaho State University | Computer Science

Isaac Griffith

CS 3321
Department of Computer Science
Idaho State University

ROAR

Idaho State University
Computer Science

Topics Covered

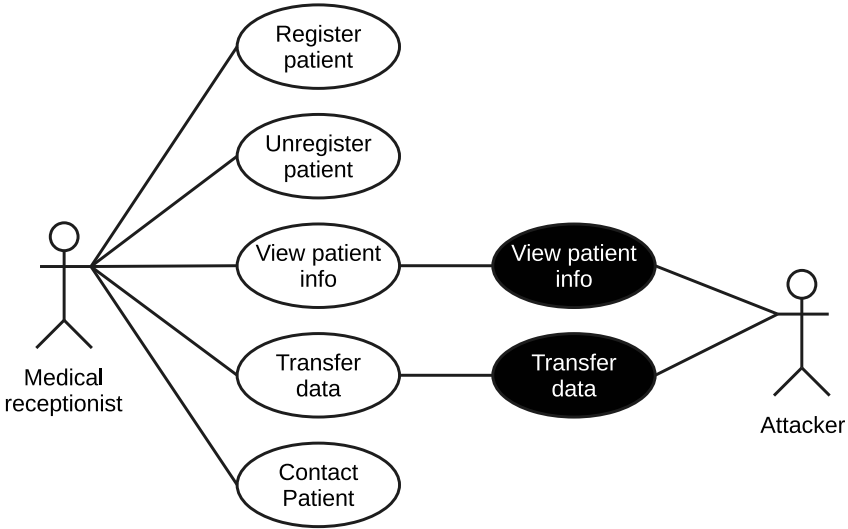
- Security requirements
- Secure systems design
- Security testing and assurance

ROAR

Misuse cases

- Misuse cases are instances of threats to a system
- Interception threats
  - Attacker gains access to an asset
- Interruption threats
  - Attacker makes part of a system unavailable
- Modification threats
  - A system asset if tampered with
- Fabrication threats
  - False information is added to a system

Misuse cases



Mentcare use case – Transfer data

Actors: Medical receptionist, Patient records system (PRS)

Description: A receptionist may transfer data from the Mentcare system to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.

Data: Patient's personal information, treatment summary.

Stimulus: User command issued by medical receptionist.

Response: Confirmation that PRS has been updated.

Comments: The receptionist must have appropriate security permissions to access the patient information and the PRS.

Idaho State University
Computer Science

Misuse case: Intercept transfer

Actors: Medical receptionist, Patient records system (PRS), Attacker

Description: A receptionist transfers data from his or her PC to the Mentcare system on the server. An attacker intercepts the data transfer and takes a copy of that data.

Data (assets): Patient's personal information, treatment summary

Attacks:

- A network monitor is added to the system and packets from the receptionist to the server are intercepted.
- A spoof server is set up between the receptionist and the database server so that receptionist believes they are interacting with the real system.

ROAR

Misuse case: Intercept transfer

Mitigations

- All networking equipment must be maintained in a locked room. Engineers accessing the equipment must be accredited.
- All data transfers between the client and server must be encrypted.
- Certificate-based client-server communication must be used

Requirements: All communications between the client and the server must use the Secure Socket Layer (SSL). The https protocol uses certificate based authentication and encryption.

Secure systems design

Secure systems design

- Security should be designed into a system – it is very difficult to make an insecure system secure after it has been designed or implemented
- Architectural design
  - how do architectural design decisions affect the security of a system?
- Good practice
  - what is accepted good practice when designing secure systems?

Design compromises

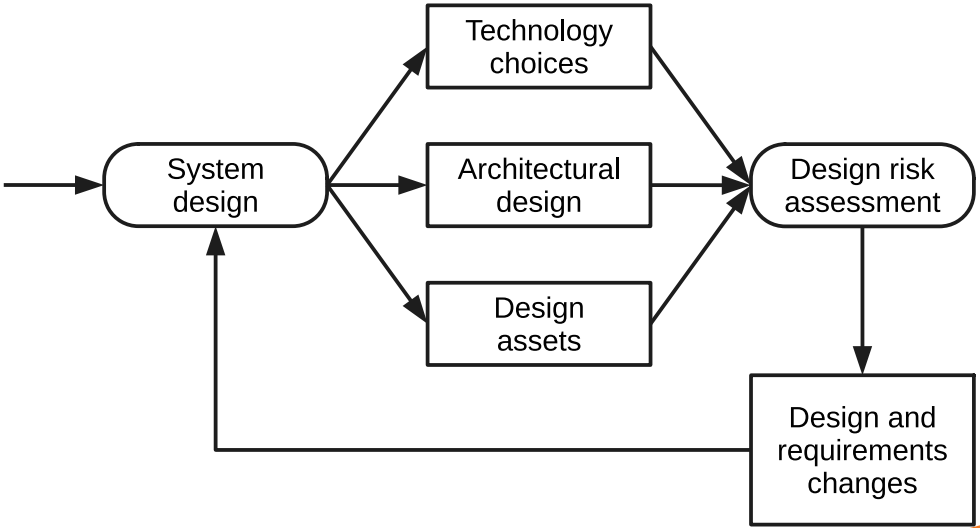
- Adding security features to a system to enhance its security affects other attributes of the system
- Performance
  - Additional security checks slow down a system so its response time or throughput may be affected
- Usability
  - Security measures may require users to remember information or require additional interactions to complete a transaction. This makes the system less usable and can frustrate system users.

Design risk assessment

- Risk assessment while the system is being developed and after it has been deployed
- More information is available - system platform, middleware and the system architecture and data organization.
- Vulnerabilities that arise from design choices may therefore be identified.

Idaho State University
Computer Science

Design and risk assessment

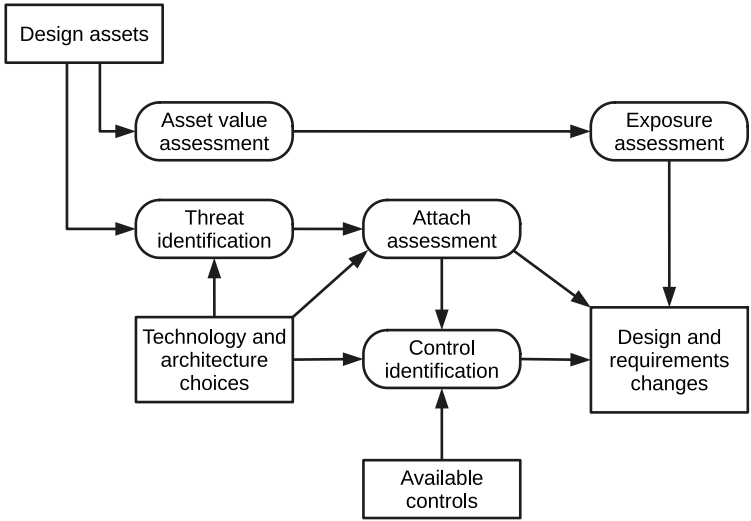


ROAR

Protection requirements

- Protection requirements may be generated when knowledge of information representation and system distribution
- Separating patient and treatment information limits the amount of information (personal patient data) that needs to be protected
- Maintaining copies of records on a local client protects against denial of service attacks on the server
  - But these may need to be encrypted

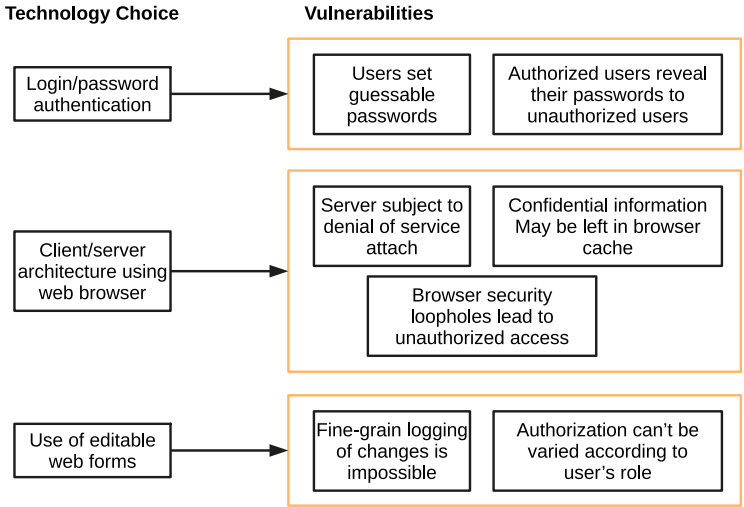
Design risk assessment



Design decisions from use of COTS

- System users authenticated using a name/password combination.
- The system architecture is client-server with clients accessing the system through a standard web browser.
- Information is presented as an editable web form.

Vulnerabilities and technology choices



Security requirements

- A password checker shall be made available and shall be run daily. Weak passwords shall be reported to system administrators.
- Access to the system shall only be allowed by approved client computers.
- All client computers shall have a single, approved web browser installed by system administrators.

Architectural design

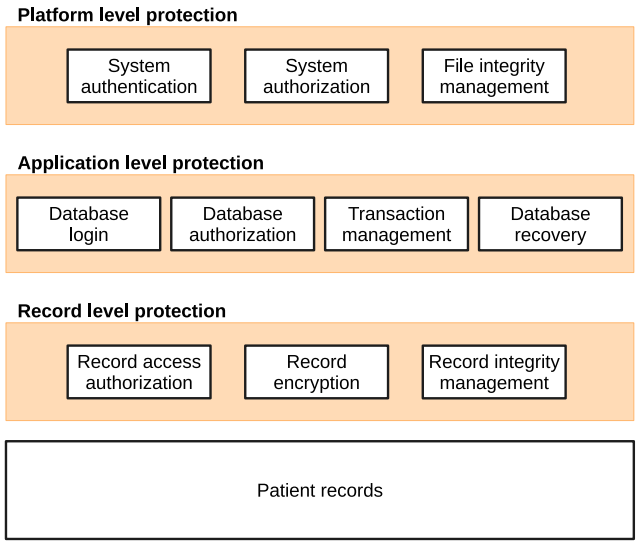
- Two fundamental issues have to be considered when designing an architecture for security.
  - Protection
    - How should the system be organized so that critical assets can be protected against external attack?
  - Distribution
    - How should system assets be distributed so that the effects of a successful attack are minimized?
- These are potentially conflicting
  - If assets are distributed, then they are more expensive to protect. If assets are protected, then usability and performance requirements may be compromised.

Protection

- Platform-level protection
  - Top-level controls on the platform on which a system runs.
- Application-level protection
  - Specific protection mechanisms built into the application itself e.g. additional password protection.
- Record-level protection
  - Protection that is invoked when access to specific information is requested
- These lead to a layered protection architecture

Idaho State University
Computer Science

A layered protection architecture



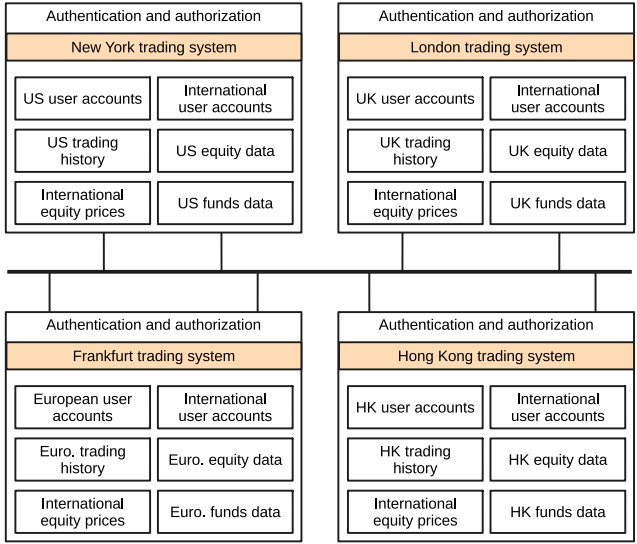
ROAR

Distribution

- Distributing assets means that attacks on one system do not necessarily lead to complete loss of system service
- Each platform has separate protection features and may be different from other platforms so that they do not share a common vulnerability
- Distribution is particularly important if the risk of denial of service attacks is high

Idaho State University
Computer Science

Equity trading system



ROAR

Security design guidelines

- Design guidelines encapsulate good practice in secure systems design
- Design guidelines serve two purposes:
  - They raise awareness of security issues in a software engineering team. Security is considered when design decisions are made.
  - They can be used as the basis of a review checklist that is applied during the system validation process.
- Design guidelines here are applicable during software specification and design

Security design guidelines

- Base security decisions on an explicit security policy
- Avoid a single point of failure
- Fail securely
- Balance security and usability
- Log user actions
- Use redundancy and diversity to reduce risk
- Specify the format of all system inputs
- Compartmentalize your assets
- Design for deployment
- Design for recoverability

Design guidelines 1-3

- Base decisions on an explicit security policy
  - Define a security policy for the organization that sets out the fundamental security requirements that should apply to all organizational systems.
- Avoid a single point of failure
  - Ensure that a security failure can only result when there is more than one failure in security procedures. For example, have password and question-based authentication.
- Fail securely
  - When systems fail, for whatever reason, ensure that sensitive information cannot be accessed by unauthorized users even although normal security procedures are unavailable.

Design guidelines 4-6

- Balance security and usability
  - Try to avoid security procedures that make the system difficult to use. Sometimes you have to accept weaker security to make the system more usable.
- Log user actions
  - Maintain a log of user actions that can be analyzed to discover who did what. If users know about such a log, they are less likely to behave in an irresponsible way.
- Use redundancy and diversity to reduce risk
  - Keep multiple copies of data and use diverse infrastructure so that an infrastructure vulnerability cannot be the single point of failure.

Design guidelines 7-10

- Specify the format of all system inputs
  - If input formats are known then you can check that all inputs are within range so that unexpected inputs don't cause problems.
- Compartmentalize your assets
  - Organize the system so that assets are in separate areas and users only have access to the information that they need rather than all system information.
- Design for deployment
  - Design the system to avoid deployment problems
- Design for recoverability
  - Design the system to simplify recoverability after a successful attack.

Secure systems programming

Secure systems programming

- Vulnerabilities are often language-specific.
  - Array bound checking is automatic in languages like Java so this is not a vulnerability that can be exploited in Java programs.
  - However, millions of programs are written in C and C++ as these allow for the development of more efficient software so simply avoiding the use of these languages is not a realistic option.
- Security vulnerabilities are closely related to program reliability.
  - Programs without array bound checking can crash so actions taken to improve program reliability can also improve system security.

Dependable programming guidelines

Dependable programming guidelines

1. Limit the visibility of information in a program
2. Check all inputs for validity
3. Provide a handler for all exceptions
4. Minimize the use of error-prone constructs
5. Provide restart capabilities
6. Check array bounds
7. Include timeouts when calling external components
8. Name all constants that represent real-world values

Security testing and assurance

Security testing

- Testing the extent to which the system can protect itself from external attacks.
- Problems with security testing
  - Security requirements are ‘shall not’ requirements i.e. they specify what should not happen. It is not usually possible to define security requirements as simple constraints that can be checked by the system.
  - The people attacking a system are intelligent and look for vulnerabilities. They can experiment to discover weaknesses and loopholes in the system.

Security validation

- Experience-based testing
  - The system is reviewed and analyzed against the types of attack that are known to the validation team.
- Penetration testing
  - A team is established whose goal is to breach the security of the system by simulating attacks on the system.
- Tool-based analysis
  - Various security tools such as password checkers are used to analyze the system in operation.
- Formal verification
  - The system is verified against a formal security specification.

Examples security checklist

Security checklist

1. Do all files that are created in the application have appropriate access permissions? The wrong access permissions may lead to these files being accessed by unauthorized users.
2. Does the system automatically terminate user sessions after a period of inactivity? Sessions that are left active may allow unauthorized access through an unattended computer.
3. If the system is written in a programming language without array bound checking, are there situations where buffer overflow may be exploited? Buffer overflow may allow attackers to send code strings to the system and then execute them.
4. If passwords are set, does the system check that passwords are 'strong'? Strong passwords consist of mixed letters, numbers, and punctuation, and are not normal dictionary entries. They are more difficult to break than simple passwords.
5. Are inputs from the system's environment always checked against an input specification? Incorrect processing of badly formed inputs is a common cause of security vulnerabilities.

Key points

- Security engineering is concerned with how to develop systems that can resist malicious attacks
- Security threats can be threats to confidentiality, integrity or availability of a system or its data
- Security risk management is concerned with assessing possible losses from attacks and deriving security requirements to minimize losses
- To specify security requirements, you should identify the assets that are to be protected and define how security techniques and technology should be used to protect these assets.

Key points

- Key issues when designing a secure systems architecture include organizing the system structure to protect key assets and distributing the system assets to minimize the losses from a successful attack.
- Security design guidelines sensitize system designers to security issues that they may not have considered. They provide a basis for creating security review checklists.
- Security validation is difficult because security requirements state what should not happen in a system, rather than what should. Furthermore, system attackers are intelligent and may have more time to probe for weaknesses than is available for security testing.

Are there any questions?