GROUP MEMBERS

Faith Muthoni – SC150/4121/2022

Lewis Mwangi – SC150/4020/2022

Michael Kamau - SC150/0120/2022

Lucy Karwitha – SC150/0034/2022

Victor Chege – SC150/5661/2021

SYSTEMS DESIGN SECURITY

Systems design security is the process of ensuring that a system is resistant to unauthorized access, modification, or destruction. It involves considering security throughout the entire system development lifecycle, from requirements gathering to deployment and maintenance.

There are a number of key principles of systems design security:

- **Defense in depth:** No single security measure is foolproof, so it is important to implement multiple layers of security. This helps to make it more difficult for attackers to compromise the system.
- **Least privilege:** Users should only be granted the access they need to perform their job duties. This helps to limit the damage that can be caused if a user's credentials are compromised.
- **Fail-safe defaults:** Systems should be designed to fail in a safe state if they encounter an error. This helps to prevent attackers from exploiting unexpected behavior.
- **Separation of duties:** Critical tasks should be divided among multiple users or systems. This helps to prevent any one user or system from having too much control.
- **Security testing:** Systems should be rigorously tested for security vulnerabilities before they are deployed. This helps to identify and remediate potential security weaknesses.

In addition to these principles, there are a number of specific security measures that can be implemented to protect systems. These include:

- **Authentication:** Users must be verified before they are granted access to the system.
- **Authorization:** Users must be authorized to perform specific actions within the system.
- **Accounting:** All user activity should be logged for auditing purposes.
- **Encryption:** Sensitive data should be encrypted to protect it from unauthorized access.
- **Firewalls:** Firewalls can be used to block unauthorized access to systems.
- **Intrusion detection/prevention systems (IDS/IPS):** IDS/IPS can be used to detect and prevent malicious activity.

By following these principles and implementing appropriate security measures, organizations can help to ensure that their systems are secure from attack.

Some specific examples of systems designs security considerations include:

- **In a web application, input validation should be used to prevent attackers from injecting malicious code into the system.**
- **In a database, access controls should be used to prevent unauthorized users from viewing or modifying sensitive data.**
- **In a network, firewalls and intrusion detection systems can be used to monitor for and block malicious traffic.**

CONTROL MEASURES

- **Threat modeling:** Identify and assess the potential threats to the system. This helps to ensure that security measures are focused on the most critical risks.
- **Secure coding practices:** Use secure coding practices to help prevent vulnerabilities from being introduced into the code. This includes things like input validation, output encoding, and error handling.
- **Security testing:** Conduct security testing throughout the development lifecycle to identify and remediate vulnerabilities. This includes things like static code analysis, dynamic application security testing, and penetration testing.
- **Least privilege:** Grant users only the access they need to perform their job duties. This helps to limit the damage that can be caused if a user's credentials are compromised.
- **Separation of duties:** Divide critical tasks among multiple users or systems. This helps to prevent any one user or system from having too much control.
- **Defense in depth:** Implement multiple layers of security. This helps to make it more difficult for attackers to compromise the system.
- **Security awareness and training:** Provide security awareness and training to users and administrators. This helps to ensure that they are aware of the security risks and how to protect the system.
- **Configuration management:** Maintain a secure configuration of the system. This includes things like keeping software up to date and applying security patches.
- **Incident response:** Have a well-defined incident response plan in place. This helps to ensure that the organization is prepared to respond to a security breach.

By implementing these control measures, organizations can help to reduce the risk of security breaches.

In addition to these general control measures, there are a number of specific control measures that can be implemented to protect system design security. These include:

- **Input validation:** Validate all user input to prevent attackers from injecting malicious code into the system.
- **Output encoding:** Encode all output to prevent cross-site scripting (XSS) attacks.
- **Error handling:** Handle errors gracefully to prevent attackers from gaining information about the system.
- **Session management:** Implement secure session management practices to prevent session hijacking attacks.
- **Cryptography:** Use cryptography to protect sensitive data.
- **Auditing and logging:** Enable auditing and logging to track user activity.
- **Monitoring:** Monitor the system for suspicious activity.

STRUCTURED DESIGN CONCEPTS

Structured design is a methodology for designing software systems that emphasizes modularity, coupling, and cohesion. The goal of structured design is to create systems that are easy to understand, maintain, and modify.

- **Modularity:** A modular system is one that is divided into independent modules. Each module has a well-defined interface and can be developed and tested independently of the other modules. This makes the system easier to understand and maintain, as changes to one module can be made without affecting the other modules.
- **Coupling:** Coupling is the degree to which modules are interconnected. A system with high coupling is one in which modules are tightly interconnected and changes to one module can have a ripple effect on other modules. A system with low coupling is one in which modules are loosely interconnected and changes to one module are unlikely to affect other modules.
- **Cohesion:** Cohesion is the degree to which the elements of a module are related. A module with high cohesion is one in which the elements are all related to a single task or function. A module with low cohesion is one in which the elements are not well-related.

Structured design uses a number of techniques to achieve modularity, coupling, and cohesion. These techniques include:

- **Functional decomposition:** Functional decomposition is the process of breaking down a system into smaller functional units. This helps to create modules that are cohesive, as each module is responsible for a single function.
- **Information hiding:** Information hiding is the principle of hiding the internal implementation details of a module from other modules. This helps to create modules that are loosely coupled, as other modules do not need to know how a module is implemented in order to use it.
- **Data flow diagrams (DFDs):** DFDs are a graphical representation of the flow of data through a system. DFDs can be used to identify the inputs, outputs, and processes of a system.

- **Structure charts:** Structure charts are a graphical representation of the hierarchical structure of a system. Structure charts can be used to show the relationships between modules.

By using these techniques, structured design can help to create systems that are modular, loosely coupled, and cohesive. This makes the systems easier to understand, maintain, and modify.

Some of the benefits of using structured design include:

- Improved understandability: Structured design helps to create systems that are easier to understand by breaking them down into smaller, more manageable modules.
- Increased maintainability: Structured design helps to create systems that are easier to maintain by making it easier to identify and isolate changes.
- Reduced complexity: Structured design helps to reduce the complexity of systems by breaking them down into smaller, more manageable modules.
- Improved quality: Structured design helps to improve the quality of systems by promoting the use of well-defined modules and interfaces.