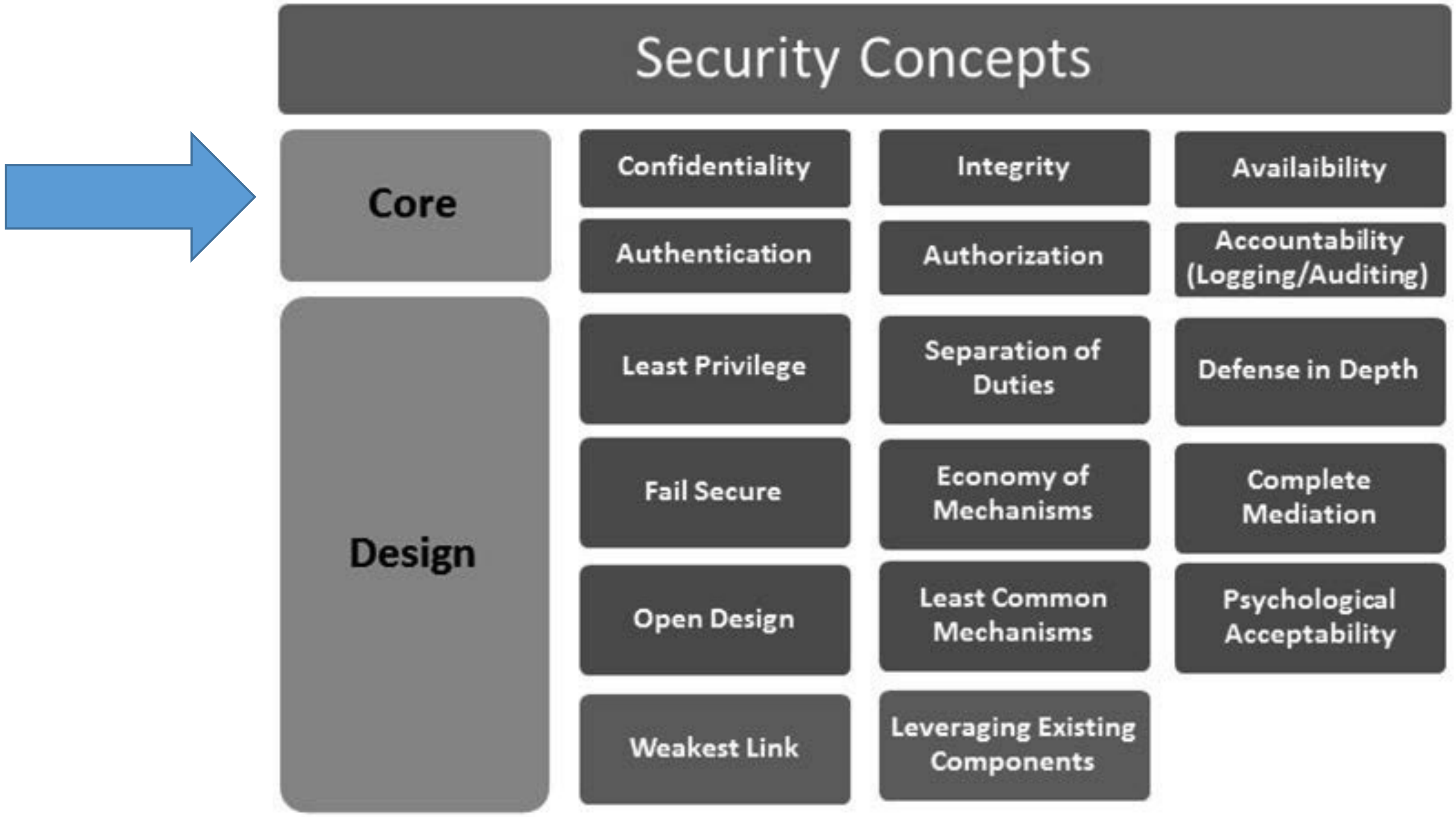
Secure Software Design and Engineering
(CY-321)

Secure Software Concepts

Dr. Zubair Ahmad

Secure Software Concepts



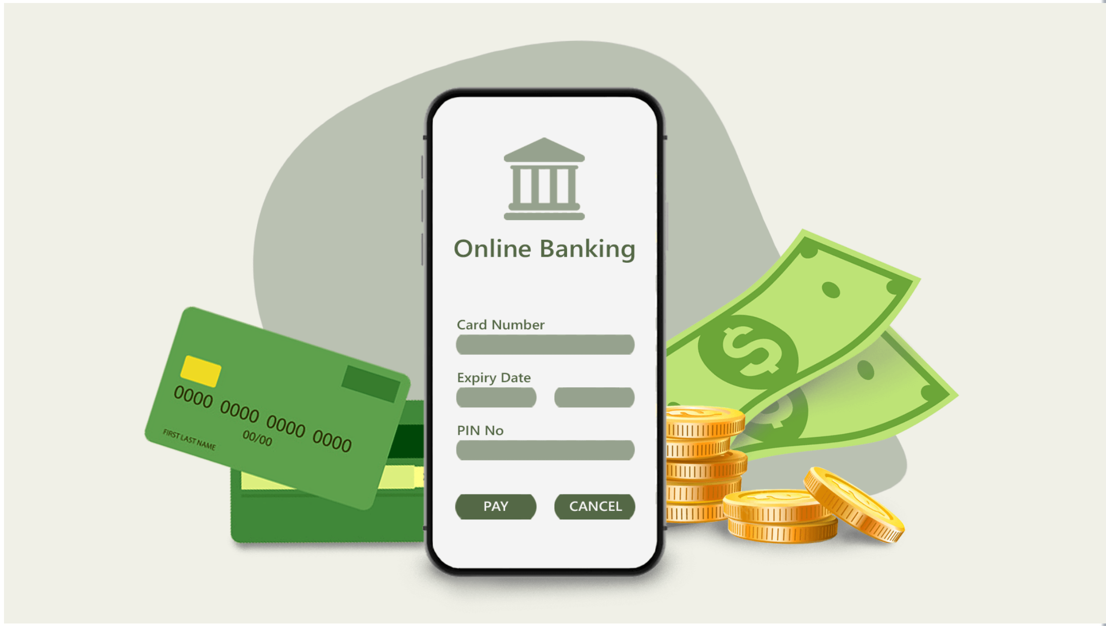
Core Secure Software Concepts (1)

Confidentiality

A security concept that has to do with protection against unauthorized Information disclosure

Banking Transactions:

Online banking uses encryption (SSL/TLS) to protect user credentials and account information from being accessed by unauthorized third parties during transactions



Core Secure Software Concepts (2)

Integrity

Protecting data against improper alteration

Data transmitted, processed and stored as accurate as the originator intented

Software Performs reliably as it was intented to

Core Secure Software Concepts (2)

Integrity (A Quick Scenario?)

Software Downloads

File Upload (Sender):
- The software provider computes a checksum (e.g., using SHA-256) of the file.
- The checksum is published on the download page.

File Download (Receiver):
- The user downloads the file and computes its checksum locally using the same algorithm.
- The user compares the locally computed checksum with the one provided by the software provider.

Integrity Verification:
- If the checksums match, the file is confirmed to be untampered.
- If the checksums differ, the file has likely been tampered with or corrupted during transmission.

Lets check the Code Examples about **confidentiality** and **integrity**

Core Secure Software Concepts (3)

Availability

Security concept that is related to the access of the software or the data or information it handles

- Downtime is minimized and that the impact upon business disruption is minimal
- “who” and “when” aspects of availability.

Core Secure Software Concepts (4)

Authentication

‘Are you whom you claim to be?”

- the person or process must be identified before it can validated or verified
- The identifying information that is supplied is also known as credentials or claims

Core Secure Software Concepts (5)

A security concept that answers the question,

'Are you whom you claim to be?"

Examples of this type of authentication include username and password, pass phrases, or a Personal Identification Number (PIN).

Knowledge
(Something that one knows)

Examples of this type of authenticati on include tokens and smart cards.

Ownership
(Something that you own or have)

biometric authentication

Characteristic
(Biometric)

Core Secure Software Concepts (6)

Authorization

The security concept in which access to objects is controlled based on the rights and privileges that are granted to the requestor by the owner of the data or system or according to a policy

- Authorization decisions are layered on top of authentication and must never precede authentication
- The requestor is referred to as the subject and the requested resource is referred to as the object

Core Secure Software Concepts (6)

Authorization (Real Life Example?)

Employee Access in a Company

Scenario:
A company's internal system has the following roles:
•**Admin**: Full access to manage users, servers, and data.
•**Manager**: Access to team data and reporting tools.
•**Employee**: Limited access to their own work data

Authorization:
After authentication, the system checks the employee's role and grants permissions accordingly:

- **Admin**: Can access all databases, edit configurations, and manage users.
- **Manager**: Can view team performance data and generate reports but cannot modify configurations.
- **Employee**: Can access their personal dashboard but no other employee’ s data or admin tools.

Core Secure Software Concepts (6)

A Quick Scenario

- You find out that the price of a product in the online store is different from the one in your physical store and you are unsure as to how this price discrepancy situation has come about.

- Upon preliminary research it is determined that the screen used to update prices of products for the online store is not tightly controlled and any authenticated user within your company can make changes to the price

Core Secure Software Concepts (6)

Auditing

Security concept in which privileged and critical businesses transactions are logged and tracked

- To build a history of events, which can be used for troubleshooting and forensic evidence

Core Secure Software Concepts (6)

Challenges in Auditing

- Performance impact
- Information overload
- Capacity limitation
- Audit log protection

Core Secure Software Concepts (7)

Non-repudiation

Addresses the deniability of actions taken by either a user or the software on behalf of the user

- **To build a history of events, which can be used for troubleshooting and forensic evidence**

- In the price change scenario, if the software had logged the price change action and the identity of the user who made that change, you can hold that individual accountable for their action and the individual has a limited opportunity to repudiate or deny their action, thereby assuring non-repudiation.

Security Concepts

Core	Confidentiality	Integrity	Availaibility
	Authentication	Authorization	Accountability (Logging/Auditing)
Design	Least Privilege	Separation of Duties	Defense in Depth
	Fail Secure	Economy of Mechanisms	Complete Mediation
	Open Design	Least Common Mechanisms	Psychological Acceptability
	Weakest Link	Leveraging Existing Components	

Design Security Concepts (1)

Least Privilege

A person or process is given only the minimum level of access rights (privileges) that is necessary for that person or process to complete an assigned operation.

ATMs (Banking)

Scenario: A person using an ATM can only withdraw money, check account balance, or deposit cash into their account. They cannot access the bank's internal systems, view another customer's account, or perform administrative functions.

Why? The ATM restricts access to only those functionalities required for the user, minimizing risk in case of a compromised system.

Design Security Concepts (2)

Separation of Duties

Successful completion of a single task is dependent upon two or more conditions that need to be met and just one of the conditions will be insufficient in completing the task by itself.

Software Development

Scenario: Developers write code, but a separate quality assurance (QA) team tests and approves it before deployment.

Why? This ensures that no single person can introduce malicious or untested code into the production environment.

Design Security Concepts (3)

Defense in Depth

Where single points of complete compromise are eliminated or mitigated by the incorporation of a series or multiple layers of security safeguards and risk-mitigation countermeasures.

Airport Security

Scenario: To board a flight, passengers go through:

- ID and boarding pass verification at the entrance.
- Security screening (metal detectors, baggage scanners).
- Additional checks.
- Passport control and visa verification (for international flights).

Why? Multiple checkpoints ensure that even if one layer misses a threat, the others may catch it.

Design Security Concepts (4)

Fail Secure

Aims to maintaining confidentiality, integrity and availability by defaulting to a secure state, rapid recovery of software resiliency upon design or implementation failure.

Bank Vaults

Scenario: If the power to an electronic vault fails, the vault automatically locks and remains secure.

Why? This prevents unauthorized individuals from accessing the contents during a failure.

Design Security Concepts (5)

Economy of Mechanisms

Likelihood of a greater number of vulnerabilities increases with the complexity of the software architectural design and code

Minimal APIs in Software

Scenario: A software application exposes only a minimal set of APIs (application programming interfaces) necessary for its operation.

Why? Reducing exposed APIs decreases the attack surface and makes the system easier to secure and maintain.

Design Security Concepts (6)

Complete Mediation

The access requests by a subject for an object is completed mediated each time, every time

File Access in Operating Systems

Scenario: When a process requests access to a file, the operating system checks the file permissions each time, even if the process accessed the file earlier

Why? This ensures that file access permissions have not been modified since the last check.

Design Security Concepts (7)

Open Design

Promotes the idea that robust security should not rely on the secrecy of a system's design

Open Source Software

Security vulnerabilities in these libraries are often discovered through community-driven review and collaboration. As a result, security flaws can be patched quickly, and the community benefits from a higher level of security

Design Security Concepts (8)

Least Common Mechanisms

Systems should avoid sharing mechanisms (e.g., software components, resources, or services) between different users or processes unless absolutely necessary

Access Control in Web Applications

In a web application, using a single set of authorization rules (e.g., access control lists, ACLs) for both admin and regular user roles could expose sensitive admin data to non-privileged users if not carefully implemented.

Implement role-based access control (RBAC), where administrators and users have separate, tailored mechanisms for accessing different resources

Design Security Concepts (8)

Psychological Acceptability

A security principle that aims at maximizing the usage and adoption of the security functionality in the software by ensuring that the security functionality is easy to use and at the same time transparent to the user

Scenario: Using a Google or Microsoft account to log in to multiple apps and websites.

Single Sign-On (SSO) Systems

Why? Reduces the need to remember multiple credentials while still enforcing authentication standards.

Design Security Concepts (8)

Weakest Link

the resiliency of your software against hacker attempts will depend heavily on the protection of its weakest components, be it the code, service or an interface.

Default or Weak Passwords

Scenario: A home router comes with a default admin password ("admin123"), which the owner never changes. An attacker easily guesses the password and accesses the network.

Why? The weak or unchanged password compromises the security of the entire network

Design Security Concepts (8)

Leveraging Existing Components

focuses on ensuring that the attack surface is not increased and no new vulnerabilities are introduced by promoting the reuse of existing software components, code and functionality.

Open-Source Libraries

Scenario: A developer needs encryption for an application. Instead of writing custom encryption algorithms, they use the **OpenSSL** library, which is robust and widely tested.

Why? Leveraging an established library ensures security and avoids errors that might arise in custom implementations.

Questions??

zubair.ahmad@giki.edu.pk

Office: G14 FCSE lobby