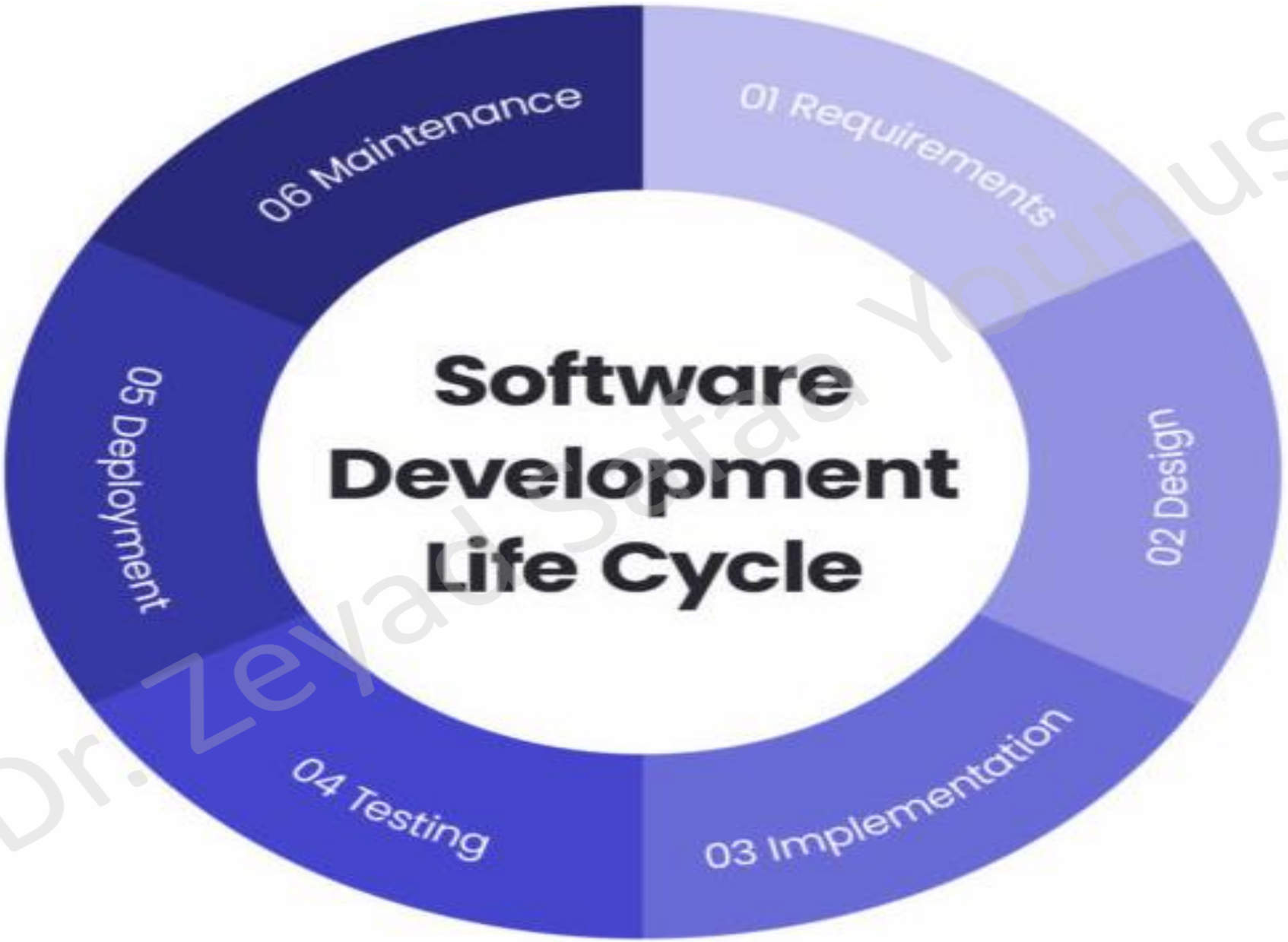
Software Security

Secure Software Development Life Cycle (SSDLC)

Dr. Zeyad Safaa Younus Saffawi

What is SSDLC

- **Secure Software Development Life Cycle** (**SSDLC**) is an approach that integrates **security practices into every stage of software development**.
- Instead of **adding security only after the software is built**, **security controls** are included **from the beginning of the development process**.
- The **goal** of SSDLC is to:
  - **Reduce security vulnerabilities**
  - **Detect security problems early**
  - **Build secure and reliable software systems**



Why SSDLC is Important

- Implementing SSDLC provides several benefits:
  - **Early detection of vulnerabilities** before deployment
  - **Reduced cost of fixing security issues**
  - **Better protection of sensitive data**
  - **Improved system reliability and trust**
  - **Compliance with security standards and regulations**

- **Fixing a security vulnerability during development** is **much cheaper and easier** than **fixing it after the system is deployed**.

Secure Software Development Lifecycle (SSDLC)

- Software security must be integrated into all stages of development:
- **Requirements Phase**
  - **Define security requirements:**
    - Identify security needs early, such as data protection, authentication, and encryption.
  - **Identify sensitive assets:**
    - Determine critical assets like user data, passwords, and financial information that require strong protection.
- **Design Phase**
  - **Threat modeling:**
    - Analyze potential threats and attack scenarios before building the system.
  - **Secure architecture design:**
    - Design the system architecture to reduce security risks and enforce access control.
- **Implementation Phase**
  - **Secure coding practices:**
    - Write code following secure coding standards to prevent common vulnerabilities.
  - **Code reviews:**
    - Review source code to identify security issues before deployment.

Secure Software Development Lifecycle (SSDLC)

- **Testing Phase**
  - **Static Application Security Testing (SAST):**
    - Analyze source code without executing it to detect security vulnerabilities.
  - **Dynamic Application Security Testing (DAST):**
    - Test the application while it is running to identify runtime vulnerabilities.
- **Deployment Phase**
  - **Secure configuration:**
    - Configure the system securely by disabling unnecessary services and applying security settings.
  - **Access control:**
    - Ensure proper authorization and assign users the minimum required privileges.
- **Maintenance Phase**
  - **Patch management:**
    - Regularly update the software to fix known vulnerabilities.
  - **Vulnerability monitoring:**
    - Continuously monitor the system to detect new security threats.

Example of SSDLC in Practice

Online Banking System

During SSDLC:

- **Requirements Phase:** define strong authentication and data protection.
- **Design Phase:** design secure architecture and access control.
- **Implementation Phase:** apply secure coding practices.
- **Testing Phase:** perform security testing (SAST and DAST).
- **Deployment Phase:** configure secure servers and databases.
- **Maintenance Phase:** update patches and monitor vulnerabilities.

This process helps ensure that the banking system remains **secure and resistant to cyber attacks**.

References

- Kohnfelder, L. (2021). *Designing secure software: A guide for developers*. O'Reilly Media.

- McGraw, G. (2006). *Software security: Building security in*. Addison-Wesley Professional.

- OWASP Foundation. (2021). *OWASP Top 10: The ten most critical web application security risks*. https://owasp.org/www-project-top-ten/