Postgraduate Certificate in Business Information Systems and Cybersecurity

Secure Software Development

Secure Software Development: Secure software development refers to the process of creating software applications with built-in security features to protect against potential threats and vulnerabilities. It involves following best practices and using secure coding techniques to ensure that the software is resistant to attacks and unauthorized access.

Key Terms and Vocabulary:

1. Threat: A potential danger that can exploit a vulnerability in a system or software application. Threats can include malware, hackers, and other malicious actors.

2. Vulnerability: A weakness in a system or software application that can be exploited by a threat actor to compromise the security of the system. Vulnerabilities can be unintentional, such as coding errors, or intentional, such as backdoors.

3. Risk: The likelihood of a threat exploiting a vulnerability and the impact it would have on the system or organization. Risks are typically assessed based on the potential damage they could cause.

4. Security Controls: Measures put in place to mitigate risks and protect against threats. Security controls can include access controls, encryption, secure coding practices, and regular security assessments.

5. Attack Surface: The sum of all possible points where a malicious actor can attempt to exploit a system. Reducing the attack surface can help improve the security of a software application.

6. Encryption: The process of encoding data in such a way that only authorized parties can access it. Encryption helps protect sensitive information from unauthorized access.

7. Authentication: The process of verifying the identity of a user or system. Authentication mechanisms can include passwords, biometrics, and two-factor authentication.

8. Authorization: The process of determining what actions a user or system is allowed to perform. Authorization controls access to resources based on the user's permissions.

9. Least Privilege: The principle of giving users or systems only the minimum level of access they need to perform their tasks. This helps reduce the risk of unauthorized access.

10. Secure Coding: The practice of writing code in a way that minimizes security vulnerabilities. Secure coding practices can help prevent common security issues such as buffer overflows and injection attacks.

11. Input Validation: The process of checking user input to ensure that it is safe and does not contain malicious code. Input validation helps prevent injection attacks and other security vulnerabilities.

12. Cross-Site Scripting (XSS): A type of vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. XSS attacks can be used to steal sensitive information or hijack user sessions.

13. SQL Injection: A type of attack that exploits vulnerabilities in SQL databases by inserting malicious SQL queries into input fields. SQL injection attacks can lead to data theft or unauthorized access to the database.

14. Secure Development Lifecycle (SDL): A structured approach to incorporating security into the software development process. The SDL includes security requirements, design, implementation, testing, and maintenance phases.

15. Threat Modeling: The process of identifying and prioritizing potential threats to a system or software application. Threat modeling helps developers understand the risks and design appropriate security controls.

16. Penetration Testing: The practice of simulating real-world attacks on a system or software application to identify vulnerabilities. Penetration testing helps organizations assess their security posture and improve defenses.

17. Secure SDLC: Secure Software Development Life Cycle (SDLC) is a methodology for designing, building, and maintaining secure software systems. It integrates security into every phase of the software development process.

18. OWASP: The Open Web Application Security Project (OWASP) is a non-profit organization that provides resources and tools for improving web application security. OWASP publishes a list of the top security risks facing web applications.

19. Security Patch: A software update that fixes a security vulnerability in a system or application. Security patches are essential for keeping software up to date and protected against known threats.

20. Secure Configuration: The process of configuring systems and software applications to minimize security risks. Secure configuration involves turning off unnecessary features, enabling security controls, and following best practices.

21. Secure Deployment: The process of securely installing and configuring software applications in a production environment. Secure deployment involves following best practices to ensure that the software is protected from attacks.

22. Secure Communication: The practice of encrypting data in transit to protect it from interception by unauthorized parties. Secure communication can be achieved using protocols such as HTTPS and SSL/TLS.

23. Compliance: The process of ensuring that software applications adhere to relevant laws, regulations, and industry standards. Compliance requirements can include data protection, privacy, and security regulations.

24. Incident Response: The process of responding to and recovering from security incidents. Incident response plans outline how to detect, analyze, and mitigate security breaches to minimize their impact.

25. Security Awareness Training: Training programs designed to educate employees and users about security best practices. Security awareness training helps raise awareness of security risks and promote a culture of security within an organization.

26. Secure Development Tools: Tools and technologies used to support secure software development practices. Secure development tools can include static analysis, dynamic analysis, and code review tools.

27. Secure Coding Guidelines: Best practices and guidelines for writing secure code. Secure coding guidelines help developers avoid common security pitfalls and vulnerabilities.

28. Secure Architecture: The design of software applications with security in mind. Secure architecture considers security requirements, threat modeling, and risk assessments to build robust and resilient systems.

29. Zero Trust Security Model: A security model that assumes no trust by default, requiring verification of every user and device trying to access resources. Zero trust security helps prevent unauthorized access and lateral movement within a network.

30. Security Operations Center (SOC): A centralized team responsible for monitoring, detecting, analyzing, and responding to security incidents. SOC teams play a critical role in maintaining the security of an organization's systems and data.

Overall, understanding and implementing secure software development practices are essential for protecting sensitive information, preventing data breaches, and maintaining the trust of users and customers. By incorporating security into every phase of the software development lifecycle, organizations can build secure and resilient applications that withstand cyber threats.