INFORMATION ASSURANCE[Y1]

ASSIGNMENT 1

University *of* Management *and* Technology

MIAN MUHAMMAD BILAL
[F2021408054]

Using Secure SDLC to develop a Secure Software

Using the Secure Software Development Lifecycle (SSDLC) process to develop a billing software application.

1. Requirements Gathering & Risk assessment

- **Functional Requirements:**
  - User accounts for managing customer information and access controls.
  - Ability to generate and send invoices with detailed billing information.
  - Secure payment processing integration for various payment methods.
  - Comprehensive reporting features for financial analysis.
- **Security Requirements:**
  - User authentication with strong password policies and multi-factor authentication (MFA).
  - Data encryption for sensitive customer information (e.g., credit card details) at rest and during transfer.
  - Access controls with role-based permissions to limit access to sensitive data.
  - Regular security audits and penetration testing to identify vulnerabilities.
  - Secure logging and monitoring for suspicious activity.

2. Threat Modeling & Secure Design

Create a model of software architecture, identifying data flow and potential attack surfaces (e.g., user login, payment processing).

- **Analyze threats like:**
  - Unauthorized access to customer accounts and financial data.
  - Man-in-the-middle attacks during payment processing.
  - SQL injection attacks on the database containing billing information.
  - Denial-of-service attacks disrupting billing operations.
- **Design the application with secure coding practices:**
  - Use secure libraries for encryption and user authentication.
  - Validate all user input to prevent malicious code injection.
  - Implement secure session management to prevent session hijacking.

3. Development & Static Analysis

Developers follow secure coding guidelines (e.g., OWASP Top 10) to prevent common vulnerabilities.
Use static analysis tools to scan the code for potential security weaknesses early in development.
Conduct code reviews with security experts to identify security flaws and coding practices that might introduce vulnerabilities.

4. Security Testing & Code review

Perform penetration testing to simulate real-world attacks and identify exploitable vulnerabilities.
Conduct dynamic analysis to detect security issues during application runtime (e.g., buffer overflows, SQL injection attacks).
Test security features like user authentication with MFA/2FA, access controls, and data encryption.
Perform security testing on the payment processing integration to ensure secure transactions.

5. Deployment and Security assessment & secure configuration

- **Deploy software to a secure production environment with:**
  - Firewalls and intrusion detection/prevention systems (IDS/IPS).
  - Secure server configuration with minimal access and patching for vulnerabilities.
  - Continuously monitor the application for suspicious activity and potential security threats.
  - Regularly update the application with security patches and fixes for newly discovered vulnerabilities.

6. Maintenance & Incident Response

Implement a process for handling security incidents, including identification, response, and recovery. Update security measures and conduct security audits periodically to ensure ongoing protection. Provide security awareness training for developers, operations teams, and customer support to maintain secure practices.