**CS 305 Project One**

Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **[1/26/2025]** | **[Shokhrukh Janobilov]** |  |

Developer

Shokhrukh Janobilov

**1. Understanding Client Needs**

* **Value of Secure Communications**: Ensuring confidentiality, integrity, and availability of sensitive financial data, such as customer savings, investments, and insurance details. Secure communications are extremely important for maintaining client trust and compliance.
* **International Transactions**: Potential international data exchanges demand abiding to data protection regulations like GDPR and ensuring secure cross-border communication protocols.
* **Governmental Restrictions**: Compliance with government-imposed standards such as PCI DSS for financial data security and encryption regulations.
* **External Threats**:
  + Injection attacks
  + Cross-Site Scripting
  + Insufficient authentication/authorization processes
  + Vulnerabilities in open-source dependencies
* **Modernization Requirements**:
  + **Open-Source Libraries**: Leveraging libraries like Spring Boot while mitigating vulnerabilities.
  + **Evolving Web Technologies**: Adopting best practices for RESTful APIs, secure coding, and emerging technologies.

**2. Areas of Security**

* **Input Validation**: Essential to prevent injection attacks by validating and sanitizing inputs
* **APIs**: Secure API interactions to prevent unauthorized access, especially in endpoints like /read and /greeting.
* **Cryptography**: Ensure proper encryption for sensitive data exchanges and storage.
* **Code Quality**: Follow secure coding practices to eliminate logic flaws and potential vulnerabilities.
* **Encapsulation**: Secure data structures, such as ensuring proper access controls in classes like customer.

**3. Manual Review**

1. **Hardcoded Database Credentials**:
   * File: DocData.java
   * Issue: Credentials for the database ("root","root") are hardcoded, posing a significant security risk.
2. **Potential SQL Injection**:
   * File: DocData.java
   * Issue: No parameterized queries or ORM usage in read\_document method.
3. **No Input Validation**:
   * File: CRUDController.java
   * Issue: Input from the business\_name parameter is not sanitized or validated.
4. **Exposed Internal Implementation**:
   * File: CRUDController.java
   * Issue: The DocData object is returned directly, exposing potential internal details.
5. **Unrestricted Endpoints**:
   * File: GreetingController.java
   * Issue: /greeting endpoint lacks authentication or authorization.
6. **Insecure Naming Conventions**:
   * File: customer.java
   * Issue: Fields account\_number and account\_balance are not encapsulated.
7. **Improper Exception Handling**:
   * File: DocData.java
   * Issue: SQLException errors are printed to the console, which can expose stack traces.
8. **Improper Data Handling**:
   * File: myDateTime.java
   * Issue: No bounds checking or validation on time fields.

**4. Static Testing**

1. **Vulnerability Code**: CVE-2018-1000180
   * **Description**: bcprov-jdk15on version 1.46 has a known vulnerability that allows attackers to exploit weaknesses in cryptographic functions.
   * **Recommended Solution**: Update to the latest version of bcprov-jdk15on (e.g., 1.70 or newer).
2. **Vulnerability Code**: CVE-2021-22119
   * **Description**: Outdated Spring Boot version (2.2.4.RELEASE) vulnerable to remote code execution.
   * **Recommended Solution**: Upgrade Spring Boot to 2.5.x or higher.
3. **Vulnerability Code**: CVE-2022-22965 (Spring4Shell)
   * **Description**: Exploits in Spring Framework that enable remote execution.
   * **Recommended Solution**: Update Spring dependencies to the latest patched versions.

**5. Mitigation Plan**

1. **Database Security**:
   * Replace hardcoded credentials in DocData.java with environment variables or secure configuration vaults.
2. **SQL Injection Prevention**:
   * Use parameterized queries or ORMs in DocData.read\_document.
3. **Input Validation**:
   * Implement robust input validation and sanitization in all endpoints, especially in CRUDController.java and GreetingController.java.
4. **Update Dependencies**:
   * Upgrade bcprov-jdk15on to version 1.70+.
   * Upgrade Spring Boot to 2.5.x or higher.
5. **Authentication and Authorization**:
   * Secure all endpoints with proper authentication and role-based authorization.
6. **Error Handling**:
   * Replace printStackTrace with proper logging methods.
   * Implement a centralized exception handler for secure error responses.
7. **Secure Coding Practices**:
   * Encapsulate sensitive fields in customer.java.
   * Conduct peer reviews for secure coding compliance.
8. **Implement Logging**:
   * Integrate security logging to monitor unauthorized access attempts and critical system events.