UTT
UNIVERSIDAD TECNOLÓGICA DE TIJUANA
GOBIERNO DE BAJA CALIFORNIA

Assignment:

Secure Coding Principles Specification

BY:

Daniel Chavez Madrigal

GROUP:

9-B

SUBJECT:

Desarrollo Móvil Integral

PROFESSOR:

Ray Brunett Parra Galaviz

Tijuana, Baja California, 15 de enero del 2025

Secure Coding Principles Specification

1. Introduction to Secure Coding Principles: Secure coding principles are guidelines and best practices designed to help developers create software that is resistant to security vulnerabilities and attacks. Implementing these principles throughout the software development lifecycle can significantly reduce the risk of security breaches

2. Key Secure Coding Principles:

- **Input Validation:** Ensure that all input is validated, sanitized, and verified before processing. This helps prevent common attacks such as SQL injection, cross-site scripting (XSS), and buffer overflows
- **Output Encoding:** Encode data before sending it to the user to prevent XSS attacks. This ensures that any potentially harmful data is rendered harmless when displayed in the browser
- **Authentication and Password Management:** Implement strong authentication mechanisms and enforce secure password policies. Use multi-factor authentication (MFA) and store passwords securely using hashing algorithms like bcrypt
- **Session Management:** Securely manage user sessions by using secure cookies, setting appropriate session timeouts, and regenerating session IDs after authentication
- **Access Control:** Implement proper access control mechanisms to ensure that users can only access resources they are authorized to. Use role-based access control (RBAC) and enforce the principle of least privilege
- **Cryptographic Practices:** Use strong encryption algorithms to protect sensitive data both in transit and at rest. Avoid using outdated or weak cryptographic algorithms
- **Error Handling and Logging:** Implement proper error handling to avoid exposing sensitive information. Log security-relevant events and ensure that logs are protected from unauthorized access
- **Data Protection:** Protect sensitive data by encrypting it and ensuring that it is only accessible to authorized users. Implement data masking and tokenization where appropriate

- **Communication Security:** Ensure secure communication between components by using protocols like HTTPS and TLS. Validate certificates and avoid using insecure communication channels

- **System Configuration:** Secure system configurations by disabling unnecessary services, applying security patches, and using secure defaults

- **Database Security:** Secure databases by using parameterized queries, restricting database access, and regularly auditing database activity

- **File Management:** Securely handle file uploads and downloads by validating file types, using secure file storage locations, and scanning files for malware

- **Memory Management:** Avoid common memory management issues such as buffer overflows and memory leaks by using safe programming practices and tools

3. Conclusion: By adhering to secure coding principles, developers can create software that is more resilient to attacks and vulnerabilities. These principles should be integrated into the software development lifecycle to ensure that security is considered at every stage of development.

References:

OWASP Secure Coding Practices - Quick Reference Guide | Secure Coding Practices | OWASP Foundation. (n.d.). https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-en/01-introduction/05-introduction

OWASP Foundation. (2010). *OWASP Secure Coding Practices Quick Reference Guide* (Version 2.1). Retrieved from https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/assets/docs/OWASP_SCP_Quick_Reference_Guide_v21.pdf