UTT
UNIVERSIDAD TECNOLÓGICA DE TIJUANA
GOBIERNO DE BAJA CALIFORNIA

TOPIC:

Secure Coding Principles Specification

PRESENTED BY:

Padilla Virgen Jorge Luis

GRUPO:

10B

SUBJECT:

Desarrollo Movil Integral

PROFESOR:

Ray Brunett Parra Galaviz

Tijuana, Baja California, January 15TH 2024

Secure Coding Principles Specification

Definition

Secure coding principles are guidelines and best practices aimed at ensuring that software is developed with security as a priority. These principles help prevent vulnerabilities that attackers can exploit, ensuring the protection of data, systems, and users.

Core Secure Coding Principles

1. **Input Validation**:
    - Ensure all user inputs are validated, sanitized, and properly constrained to prevent injection attacks like SQL injection, cross-site scripting (XSS), and buffer overflows.
2. **Least Privilege**:
    - Limit access rights for users and applications to the minimum necessary to perform their tasks, reducing the risk of unauthorized actions.
3. **Secure Authentication and Authorization**:
    - Use robust authentication mechanisms (e.g., multi-factor authentication).
    - Implement role-based access control (RBAC) to restrict access based on user roles.
4. **Error Handling and Logging**:
    - Avoid exposing sensitive information in error messages.
    - Implement logging to monitor activities but ensure logs do not store sensitive data like passwords or encryption keys.

5. **Data Encryption**:
    - Encrypt sensitive data both in transit and at rest using modern cryptographic algorithms.
6. **Avoid Hardcoding Secrets**:
    - Do not hardcode sensitive information like API keys, passwords, or credentials in the source code. Use secure secret management tools.
7. **Secure Session Management**:
    - Use secure cookies and implement measures like session expiration and regeneration to prevent session hijacking.
8. **Defense in Depth**:
    - Layer security controls to provide multiple lines of defense against threats.
9. **Secure Configuration**:
    - Ensure that default configurations of software, servers, and frameworks are changed to secure settings.
    - Disable unused features and components to reduce the attack surface.
10. **Validation of Dependencies**:
    - Regularly update third-party libraries and frameworks to patch known vulnerabilities.
    - Use tools to check for vulnerabilities in dependencies (e.g., OWASP Dependency-Check, Snyk).
11. **Avoid Security Through Obscurity**:
    - Do not rely on hiding implementation details as the sole security measure; combine with robust security mechanisms.

12. **Secure Code Reviews**:
    - Regularly review and test code for vulnerabilities using static analysis tools and peer reviews.

Applicable Standards and Frameworks

1. **OWASP Secure Coding Practices**:
    - A comprehensive checklist covering secure coding techniques to mitigate common vulnerabilities.
2. **ISO/IEC 27001**:
    - Provides guidelines for implementing information security management systems.
3. **NIST SP 800-53**:
    - A framework for security and privacy controls for federal information systems.
4. **CERT Secure Coding Standards**:
    - Language-specific secure coding guidelines (e.g., for C, C++, Java) to prevent common programming errors.

Common Vulnerabilities Addressed

1. **Injection Flaws** (e.g., SQL injection, command injection).
2. **Cross-Site Scripting (XSS)**.
3. **Cross-Site Request Forgery (CSRF)**.
4. **Buffer Overflows**.
5. **Insecure Deserialization**.
6. **Broken Authentication and Session Management**.

Benefits of Secure Coding

1. **Enhanced Security**:
    - Reduces vulnerabilities that attackers can exploit.
2. **Regulatory Compliance**:
    - Meets legal and industry-specific security requirements.
3. **Cost Savings**:
    - Mitigates costs associated with breach recovery and reputation damage.
4. **Improved Reliability**:
    - Ensures robust and resilient software.

Challenges

1. **Developer Awareness**:
    - Requires training and awareness of security risks.
2. **Balancing Security and Functionality**:
    - Incorporating security without hindering usability.
3. **Evolving Threat Landscape**:
    - Staying updated with emerging threats and vulnerabilities.