UTT
UNIVERSIDAD TECNOLÓGICA DE TIJUANA
GOBIERNO DE BAJA CALIFORNIA

TEMA:

Secure Coding Principles Specification

PRESENTADO POR:

Hernández Miranda Rafael Francisco

GRUPO:

10B

MATERIA:

Desarrollo Móvil Integral

PROFESOR:

Ray Brunett Parra Galaviz

FECHA:

15/01/2025.

Secure Coding Principles Specification

Secure coding principles are essential guidelines that help developers write code that is robust against vulnerabilities and threats. Here's a summary of key principles and best practices for secure coding:

1. Input Validation

- Always validate input data to ensure it meets expected formats and constraints. This helps prevent injection attacks and ensures data integrity.

2. Error Handling

- Implement proper error handling to avoid exposing sensitive information through error messages. Errors should be logged appropriately without revealing system details to the user.

3. Principle of Least Privilege

- Ensure that code runs with the minimum privileges necessary. This limits the potential damage in case of a security breach.

4. Secure Access Control

- Implement strict access controls to sensitive data and functionalities, ensuring that only authorized users can access or modify them.

5. Use of Secure Libraries and Frameworks

- Utilize well-maintained libraries and frameworks that follow secure coding practices, reducing the risk of vulnerabilities from third-party code.

6. Regular Code Reviews and Static Analysis

- Conduct regular code reviews and use static analysis tools to identify potential security flaws early in the development process.

7. Avoid Hardcoding Secrets

- Never hardcode sensitive information such as passwords or API keys in the source code. Use secure storage solutions instead.

8. Data Encryption

- Encrypt sensitive data both at rest and in transit to protect it from unauthorized access.

9. Keep Software Updated

- Regularly update libraries, frameworks, and dependencies to patch known vulnerabilities.

10. Follow Industry Standards

- Adhere to established coding standards such as OWASP Top Ten, CERT, or MISRA C for guidance on secure coding practices tailored to specific industries.

Conclusion

Implementing secure coding principles is vital for developing resilient software systems. Key practices include input validation, robust error handling, strict authentication, access control based on least privilege, effective logging, and regular quality assurance checks. Continuous developer training and integration of these principles throughout the SDLC enhance security posture against evolving threats.

Reference

- University of Michigan. (n.d.). Best practices for secure coding. Retrieved January 16, 2025, from https://safecomputing.umich.edu/protect-the-u/protect-your-unit/secure-coding/best-practices
- KirkpatrickPrice. (2023). 8 best secure coding practices. Retrieved January 16, 2025, from https://kirkpatrickprice.com/blog/secure-coding-best-practices/
- Jit.io. (n.d.). 7 principles of secure design in software development. Retrieved January 16, 2025, from https://www.jit.io/resources/app-security/secure-design-principles
- University of California, Berkeley. (n.d.). Secure coding practice guidelines. Retrieved January 16, 2025, from https://security.berkeley.edu/secure-coding-practice-guidelines
- OWASP Foundation. (n.d.). Secure coding practices checklist. Retrieved January 16, 2025, from https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-en/02-checklist/05-checklist
- ISMS.online. (2022). ISO 27002:2022 control 8.28, secure coding. Retrieved January 16, 2025, from https://www.isms.online/iso-27002/control-8-28-secure-coding/
- National Cyber Security Centre (NCSC). (n.d.). Secure development principles. Retrieved January 16, 2025, from https://www.ncsc.gov.uk/collection/developers-collection/principles