UTT
UNIVERSIDAD TECNOLÓGICA DE TIJUANA
GOBIERNO DE BAJA CALIFORNIA

TEMA

Secure Coding Principles Specification

PRESENTADO POR

Perez Bello Vanory Esperanza

GRUPO

10° B

MATERIA

Desarrollo Movil Integral

PROFESOR

Ray Brunett Parra Galaviz

Tijuana, Baja California, 15 Enero del 2025

Especificación de Principios de Codificación Segura

La codificación segura se refiere a la práctica de desarrollar software con un enfoque específico en la seguridad, minimizando las vulnerabilidades que los atacantes podrían explotar. Este enfoque implica el uso de técnicas, estándares y metodologías probadas para garantizar que el software sea resistente frente a amenazas actuales y futuras. Los principios de codificación segura se fundamentan en prevenir errores que puedan comprometer la confidencialidad, integridad y disponibilidad de los sistemas.

Principios Fundamentales de Codificación Segura

1. **Validación de Entradas**

    - Las entradas provenientes de usuarios, sistemas externos o cualquier fuente no controlada deben considerarse no confiables.
    - La validación de entradas implica verificar que los datos sean del tipo, tamaño y formato esperados antes de procesarlos.
    - Técnicas como la sanitización y normalización previenen ataques de inyección SQL, desbordamiento de búfer, y otras amenazas relacionadas.

2. **Codificación y Sanitización de Salidas**

    - Antes de presentar datos al usuario o enviarlos a otros sistemas, deben ser codificados para evitar interpretaciones inesperadas.
    - Esto es crucial para prevenir ataques de Cross-Site Scripting (XSS), asegurando que los datos sean tratados como texto simple y no como código ejecutable.

3. **Control de Acceso y Privilegios Mínimos**

- Implementar políticas de control de acceso asegura que los usuarios y procesos tengan solo los privilegios necesarios para realizar sus funciones.
- El principio de privilegio mínimo limita los riesgos al restringir el acceso a los recursos críticos.

4. **Gestión Segura de Sesiones**

    - Las sesiones de usuario deben gestionarse de manera segura, utilizando identificadores únicos y aleatorios.
    - Es importante proteger las cookies de sesión con atributos como *Secure* y *HttpOnly*, además de implementar mecanismos de expiración.

5. **Autenticación y Gestión de Credenciales**

    Utilizar contraseñas seguras y técnicas como hash y salting para almacenarlas.

    Se deben aplicar factores de autenticación múltiples (MFA) y proteger contra ataques como fuerza bruta y relleno de credenciales (*credential stuffing*).

6. **Cifrado y Protección de Datos Sensibles**

    - Los datos deben protegerse tanto en tránsito como en reposo mediante el uso de algoritmos criptográficos robustos como AES y RSA.
    - Las claves de cifrado deben gestionarse con herramientas especializadas y mantenerse fuera del código fuente.

7. **Gestión de Errores**

   Los errores y excepciones deben ser manejados de manera que no revelen detalles técnicos a los usuarios.

   Los registros de errores deben ser informativos para los desarrolladores, pero no deben incluir información sensible como claves, contraseñas o detalles del sistema.

8. **Defensa en Profundidad**

   - Este principio sugiere implementar múltiples capas de seguridad para proteger los sistemas.
   - Por ejemplo, incluso si una capa (como la validación de entrada) falla, otras (como el cifrado de datos) proporcionarán protección adicional.

9. **Evitar Dependencias Inseguras**

   - Asegurarse de que las bibliotecas, frameworks y herramientas de terceros utilizados en el desarrollo estén actualizados y libres de vulnerabilidades conocidas.
   - Utilizar herramientas de análisis de composición de software (SCA) para identificar riesgos en las dependencias.

10. **Pruebas de Seguridad Continua**

- Las pruebas de seguridad no deben limitarse al final del ciclo de desarrollo.
- Realizar análisis estático y dinámico del código, pruebas de penetración, y revisiones manuales en cada etapa del desarrollo asegura la identificación temprana de vulnerabilidades.

Importancia de los Principios de Codificación Segura

El desarrollo de software inseguro puede tener consecuencias graves, como pérdida de datos, daños a la reputación, multas regulatorias y costos asociados con la recuperación de incidentes de seguridad. La codificación segura ayuda a prevenir estas situaciones al identificar y abordar vulnerabilidades desde la fase de diseño. Además, el cumplimiento de estándares como OWASP, NIST, y ISO/IEC 27001 refuerza la postura de seguridad de las organizaciones.

Desafíos Comunes y Mejores Prácticas

1. **Falta de Conciencia sobre Seguridad**
    - Es necesario capacitar a los desarrolladores en seguridad para que comprendan y apliquen prácticas de codificación segura.
    - Se pueden implementar talleres, guías y herramientas como IDEs con análisis de seguridad integrado.
2. **Complejidad en los Sistemas**
    - A medida que los sistemas se vuelven más complejos, también lo hacen las amenazas.
    - Una arquitectura bien diseñada que siga principios de separación de responsabilidades y modularidad reduce el riesgo.
3. **Integración con el Desarrollo Ágil y DevOps**
    - Incorporar seguridad en DevOps (*DevSecOps*) asegura que la seguridad no sea un impedimento, sino una parte integral del proceso de desarrollo.

Conclusión

Los principios de codificación segura son esenciales para crear software confiable y resistente frente a las crecientes amenazas cibernéticas. Adoptar estas prácticas no solo protege los activos de una organización, sino que también construye confianza en los usuarios y cumple con las normativas de seguridad. Integrar estos principios en el ciclo de desarrollo de software debe ser una prioridad para cualquier equipo de desarrollo moderno.

Referencias Bibliográficas

OWASP Foundation. (s.f.). *Secure Coding Practices Checklist*. Recuperado de https://owasp.org

Information Security Office. (s.f.). *Secure Coding Practice Guidelines*. Recuperado de https://security.berkeley.edu

Perforce Software. (s.f.). *What Are Secure Coding Standards?*. Recuperado de https://www.perforce.com

Jit.io. (s.f.). *7 Principles of Secure Design in Software Development*. Recuperado de https://www.jit.io

Snyk. (s.f.). *The 3 Pillars of Implementing Secure Coding Standards*. Recuperado de https://snyk.io