UTT
UNIVERSIDAD TECNOLÓGICA DE TIJUANA
GOBIERNO DE BAJA CALIFORNIA

TEMA:

Secure Coding Principles Specification

PRESENTADO POR:

Sanchez Zamudio Guadalupe

GRUPO:

10B

MATERIA:

Desarrollo Móvil Integral

PROFESOR:

Ray Brunett Parra Galaviz

Tijuana, Baja California, 15 de enero del 2025

Los principios de codificación segura son un conjunto de prácticas y lineamientos diseñados para garantizar que el software desarrollado sea resistente a vulnerabilidades de seguridad.

1. Validación de entradas

- Todas las entradas deben considerarse no confiables hasta que sean validadas.
- Utilizar listas blancas para aceptar únicamente datos esperados y válidos.
- Limitar el tamaño, formato y tipo de datos recibidos.
- Validar tanto en el lado del cliente como en el servidor.

2. Control de acceso

- Implementar controles de acceso basados en roles o atributos.
- Verificar permisos antes de permitir el acceso a recursos o funciones.
- Asegurarse de que las APIs y rutas estén protegidas con autenticación y autorización.
- Prevenir el acceso no autorizado a datos sensibles.

3. Gestión de sesiones

- Usar identificadores de sesión únicos y regenerarlos después de autenticación.
- Configurar las sesiones para que expiren automáticamente tras un período de inactividad.
- Asegurarse de que las cookies de sesión sean seguras, de solo lectura y protegidas contra ataques como secuestro de sesiones.

4. Uso de criptografía segura

- Proteger contraseñas y datos sensibles con algoritmos criptográficos robustos.
- Evitar almacenar contraseñas en texto plano.
- Usar algoritmos y bibliotecas criptográficas actualizadas y evitar las obsoletas.
- Cifrar los datos tanto en tránsito como en reposo.

5. Manejo seguro de errores

- Evitar mostrar detalles técnicos de errores a los usuarios finales.
- Configurar mensajes de error genéricos para no exponer información sensible.
- Registrar los errores en sistemas seguros para facilitar el diagnóstico y la auditoría.

6. Protección contra ataques comunes

- **Cross-Site Scripting (XSS)**: Escapar y validar las entradas y salidas en aplicaciones web.
- **Cross-Site Request Forgery (CSRF)**: Implementar tokens únicos para prevenir solicitudes no autorizadas.
- **SQL Injection**: Utilizar consultas parametrizadas o un ORM para evitar la manipulación de bases de datos.
- **Inyección de comandos**: Validar entradas antes de usarlas en llamadas a sistemas operativos.

7. Principio del menor privilegio

- Otorgar a los usuarios y sistemas únicamente los permisos necesarios para realizar sus tareas.
- Restringir accesos administrativos y privilegios elevados.
- Asegurarse de que los servicios y procesos tengan permisos mínimos.

8. Actualización y mantenimiento

- Mantener actualizado el software y las bibliotecas utilizadas.
- Monitorear y aplicar parches de seguridad rápidamente.
- Revisar periódicamente el código para detectar vulnerabilidades.

9. Seguridad en el ciclo de vida

- Incorporar prácticas de codificación segura desde el diseño hasta la implementación.
- Realizar pruebas de seguridad estáticas y dinámicas en el código.
- Implementar revisiones de código para identificar problemas antes del despliegue.

10. Registro y monitoreo

- Registrar eventos relevantes, como intentos de acceso fallidos o cambios en datos críticos.
- Monitorear los registros para identificar y responder a posibles incidentes de seguridad.
- Asegurarse de que los registros no expongan datos sensibles.