Top 10 Controles Proactivos de Seguridad (OWASP)

1. Define Security Requirements
    - Establece requisitos de seguridad desde el principio del proyecto.
    - Ejemplo: autenticación, control de acceso, protección de datos sensibles.
2. Leverage Security Frameworks and Libraries
    - Usa bibliotecas bien mantenidas para evitar reinventar la rueda (ej. para cifrado, autenticación).
    - Reduce riesgos por errores propios al crear lógica de seguridad manual.
3. Secure Database Access
    - Evita inyecciones SQL utilizando ORM o consultas parametrizadas.
    - Principio del menor privilegio: la app no debe tener acceso total a la base.
4. Encode and Escape Data
    - Protege contra XSS y otras inyecciones asegurándote de codificar datos antes de mostrarlos en HTML, JavaScript, etc.
5. Validate All Inputs

- Valida y sanitiza todo lo que entra (formularios, URLs, headers).
- Asegura que los datos cumplan con formatos esperados.

6. Implement Digital Identity

- Usa autenticación segura: contraseñas fuertes, autenticación multifactor (MFA), protocolos modernos como OAuth2/OpenID Connect.

7. Enforce Access Controls

- Define qué puede hacer cada usuario y asegúrate de verificarlo en cada acción.
- No confíes solo en la interfaz para ocultar funciones.

8. Protect Data Everywhere

- Cifra datos sensibles tanto en tránsito (HTTPS) como en reposo.
- Evita exponer datos personales o credenciales.

9. Implement Security Logging and Monitoring

- Registra eventos clave (inicio de sesión, errores, accesos no autorizados).
- Usa monitoreo para detectar incidentes rápidamente.

10. Handle All Errors and Exceptions

- No muestres errores detallados al usuario.

- Manejá excepciones para evitar fugas de información.

📌 Apéndices y extras:

- Incluye referencias a herramientas OWASP como ASVS, Top 10, Cheat Sheets.
- Presenta ejemplos en varios lenguajes (Java, Python, .NET).
- Recomendaciones para desarrolladores, testers y equipos de DevSecOps.

🛡️ Guía de los 10 Controles Proactivos de Seguridad (OWASP 2024)

1. 🗒️ Definir Requisitos de Seguridad

- Establecé requisitos de seguridad desde el inicio del proyecto.
- Pensá en: control de acceso, cifrado, privacidad, protección ante ataques comunes.
- Usá estándares como OWASP ASVS (Application Security Verification Standard).

---

2. 📚 Usar Frameworks y Librerías de Seguridad

- Evitá escribir tu propia lógica de seguridad.
- Preferí librerías reconocidas (por ejemplo: Spring Security, ASP.NET Identity).
- Mantenelas actualizadas para evitar vulnerabilidades.

---

3. 🗃️ Asegurar el Acceso a la Base de Datos

- Usá consultas parametrizadas o ORM para prevenir inyección SQL.
- Aplicá el principio de menor privilegio: la app solo debe acceder a lo necesario.

---

4. 🔐 Codificar y Escapar los Datos

- Protegé contra ataques XSS (Cross-Site Scripting) y otros de inyección.
- Codificá los datos antes de mostrarlos en HTML, JavaScript, etc.

---

5. 🛂 Validar Todas las Entradas

- Toda entrada del usuario debe validarse y/o sanearse.
- Usá validaciones de tipo, formato, longitud, etc.
- Nunca confíes en el cliente (validá también en el servidor).

---

6. 👤 Implementar Identidad Digital

- Usá autenticación segura: contraseñas fuertes, MFA, sesiones protegidas.
- Aplicá estándares como OAuth 2.0, OpenID Connect.

---

7. 🧱 Aplicar Control de Accesos

- Definí roles y permisos claros.
- Verificá que los usuarios solo puedan realizar acciones autorizadas.
- No dependas solo de la interfaz para restringir funciones.

---

8. 🧬 Proteger los Datos en Todo Momento

- Cifrá los datos sensibles en tránsito (HTTPS) y en reposo (base de datos).
- Nunca almacenes contraseñas en texto plano.

---

9. 📜 Registrar y Monitorear Seguridad

- Registrá eventos clave como inicios de sesión, errores, accesos fallidos.
- Usá sistemas de monitoreo para detectar actividad sospechosa.

---

10. 🚫 Manejar Errores y Excepciones

- Nunca muestres errores técnicos detallados al usuario.
- Manejá correctamente excepciones para evitar fugas de información.

📎 Recomendaciones Extra:

- Usá las guías de OWASP (Top 10, ASVS, Cheat Sheets).
- Automatizá pruebas de seguridad en el ciclo de desarrollo (DevSecOps).
- Fomentá la capacitación en seguridad del equipo de desarrollo.