A06:2025

Diseño Inseguro

OWASP Top 10 — 2025

Fundamentos de Ciberseguridad — IES Los Sauces — Curso 2025/2026

Enrique Nieto Lorenzo

¿Qué es el Diseño Inseguro?

Controles de seguridad que nunca se planificaron en la arquitectura del sistema.

Fallo de Diseño

La seguridad nunca se consideró.

No se corrige con código perfecto.

Requiere rediseño completo.

Defecto de Implementación

El diseño es seguro pero se programó mal.

Se corrige arreglando el código.

¿Qué consecuencias tiene?

Sin modelado de amenazas	→ Vulnerabilidades no detectadas hasta producción
Sin separación de roles	→ Elevación de privilegios trivial
Lógica de negocio sin validar	→ Abuso de funcionalidades
Sin límite de intentos	→ Fuerza bruta exitosa
Mensajes de error diferenciados	→ Enumeración de usuarios válidos

Las consecuencias son persistentes: no se parchean, se rediseñan.

¿Cómo prevenirlo? Principios generales

Modelado de Amenazas

Analizar qué activos existen, quién los atacaría y cómo, antes de codificar.

Historias de Abuso

Por cada historia de usuario, escribir cómo un atacante intentaría romperla.

Ciclo de Desarrollo Seguro (SDLC)

Integrar seguridad en cada fase del desarrollo, no solo al final.

Patrones de Diseño Seguros

Usar componentes reutilizables y probados en lugar de soluciones ad-hoc.

¿Cómo lo previene nuestra aplicación DWES?

Encapsulación (clases POPO)

Atributos privados en Usuario, Departamento, Cuestion. Ninguna capa externa accede directamente a datos sensibles.

Clases PDO — punto único de acceso a BD

Todo pasa por DBPDO::ejecutaConsulta() con sentencias preparadas. Inyección SQL eliminada por diseño.

Front Controller (index.php + confAPP.php)

Verificación de sesión y roles en un único punto. Páginas privadas por defecto.

ValidaForms y ErrorApp

Validación unificada y control de fuga de información en clases transversales del core.

Testeo sobre nuestra aplicación

Prueba	Qué hacer	Vulnerable si...
Fuerza bruta	Enviar +5 logins fallidos seguidos	No bloquea ni limita
Enumeración de usuarios	Comparar mensajes con usuario real vs inventado	Los mensajes son diferentes
Session fixation	Anotar PHPSESSID antes y después del login	El ID no cambia
Timeout de sesión	Dejar inactiva la sesión 30 min	No expira
Control de acceso	Acceder como usuario normal a URL de admin	La página carga

Grado de afectación

Diseño Robusto

- OOP en el modelo
- Arquitectura MVC
- Front Controller centralizado
- ValidaForms (validación unificada)
- ErrorApp (control de errores)

Carencias Puntuales

- Falta rate limiting en login
- Carencia de session_regenerate_id
- Mensajes de error diferenciados en registro

Las decisiones de diseño tomadas al inicio son las que realmente protegen la aplicación. Las carencias son añadibles sin rediseñar.

«Un diseño seguro no se improvisa:

se planifica desde el primer diagrama de clases.»

¡Gracias!

¿Preguntas?