

OWASP Top 10

Ingeniería de Software 2 (CC3091) - 2026

OWASP Top 10

Semestre 02, 2026

Introducción

La seguridad no es una fase que se agrega al final del proyecto, es una propiedad que se construye durante todo el ciclo de desarrollo.

El OWASP Top 10 es el documento de referencia más usado en la industria para entender los riesgos de seguridad más críticos en aplicaciones web.

No es una lista de vulnerabilidades sueltas, es una lista de causas raíz: categorías de errores de diseño, construcción y configuración que se repiten en miles de aplicaciones.

OWASP

OWASP (Open Worldwide Application Security Project) es una fundación sin fines de lucro dedicada a mejorar la seguridad del software.

Todo su material es abierto y gratuito, producido por una comunidad global de desarrolladores, auditores e investigadores.

Además del Top 10, mantiene proyectos como ZAP (escáner de seguridad), ASVS (estándar de verificación), Cheat Sheet Series y Dependency-Check.

El Top 10 como estándar de referencia

El Top 10 es un documento de concientización, no una certificación ni una lista de cumplimiento.

Su valor está en dar un lenguaje común entre desarrolladores, arquitectos, QA y seguridad para hablar de riesgo.

Muchos marcos regulatorios y contratos de software lo citan como línea base mínima de seguridad.

La edición vigente es OWASP Top 10:2025, publicada en su versión final en enero de 2026.

Cómo se construye la lista

La lista no se arma por opinión, se arma con datos de la industria. Para leer esos datos hacen falta tres siglas.

- **CWE** (Common Weakness Enumeration): catálogo público de tipos de error de programación, como no validar la entrada del usuario o guardar contraseñas sin cifrar.
- **CVE** (Common Vulnerabilities and Exposures): registro de vulnerabilidades concretas encontradas en productos reales, cada una con su identificador.
- **CVSS** (Common Vulnerability Scoring System): puntaje de 0 a 10 que mide qué tan grave es una vulnerabilidad registrada.

Con ese vocabulario, la construcción de la lista funciona así:

- **Fuente de datos:** 13 organizaciones aportaron los resultados de sus pruebas sobre más de 2.8 millones de aplicaciones.
- **Agrupación:** cada categoría del Top 10 agrupa varios CWE emparentados, 248 en total.
- **Prevalencia:** porcentaje de aplicaciones probadas donde se encontró al menos un error de esa categoría. Las categorías más prevalentes suben de posición.
- **Gravedad:** sale de los CVE ya reportados y de sus puntajes CVSS.
- **Encuesta comunitaria:** ocho categorías salen de los datos y dos se eligen por encuesta, para capturar riesgos emergentes que las herramientas todavía no detectan a escala.

El criterio de agrupación es la causa raíz sobre el síntoma: se clasifica el error que origina el problema, no la forma en que se manifiesta.

Seguridad en el ciclo de desarrollo

Corregir un defecto de seguridad en producción cuesta mucho más que evitarlo en diseño.

Por eso la práctica actual es desplazar la seguridad hacia la izquierda del ciclo (shift left), integrándola en requerimientos, diseño, construcción y despliegue.

- **Requerimientos:** incluir requerimientos de seguridad y casos de abuso junto a los casos de uso.
- **Diseño:** modelado de amenazas para anticipar cómo puede fallar el sistema.
- **Construcción:** revisiones de código, análisis estático y bibliotecas seguras por defecto.

- **Pruebas:** análisis dinámico, pruebas de penetración y pruebas de regresión de seguridad.
- **Despliegue y operación:** configuración endurecida, registro de eventos y monitoreo continuo.

La lista OWASP Top 10:2025

- **A01** Broken Access Control — Control de acceso roto
- **A02** Security Misconfiguration — Configuración de seguridad incorrecta
- **A03** Software Supply Chain Failures — Fallas en la cadena de suministro
- **A04** Cryptographic Failures — Fallas criptográficas
- **A05** Injection — Inyección
- **A06** Insecure Design — Diseño inseguro
- **A07** Authentication Failures — Fallas de autenticación
- **A08** Software or Data Integrity Failures — Fallas de integridad de software o datos
- **A09** Security Logging and Alerting Failures — Fallas de registro y alerta
- **A10** Mishandling of Exceptional Conditions — Manejo incorrecto de condiciones excepcionales

A01 — Broken Access Control

El control de acceso hace cumplir la política de que un usuario no puede actuar fuera de los permisos que le corresponden.

Cuando falla, un atacante accede a datos ajenos, modifica información o ejecuta funciones de negocio que no le corresponden.

Es la categoría número uno desde 2021 y en 2025 absorbió también el SSRF (Server-Side Request Forgery).

Fallas comunes:

- Violar el principio de menor privilegio, permitiendo por defecto en lugar de denegar por defecto.
- Referencias directas a objetos inseguras, donde cambiar un identificador en la URL muestra el registro de otro usuario.
- Falta de control de acceso en métodos de API como POST, PUT y DELETE.
- Elevación de privilegios manipulando tokens JWT, cookies o parámetros ocultos.
- Navegación forzada a páginas administrativas sin autenticación.

Cómo prevenirlo:

- Denegar por defecto todo recurso que no sea explícitamente público.
- Implementar el control de acceso del lado del servidor, nunca en el código del cliente.
- Validar la propiedad del registro en el modelo de dominio, no solo el rol del usuario.
- Registrar los intentos fallidos y aplicar límites de tasa en las API.
- Invalidar la sesión al cerrar sesión y usar tokens de vida corta.

A02 — Security Misconfiguration

El software puede estar bien programado y aun así ser inseguro por cómo está configurado.

Subió del puesto 5 al puesto 2, impulsada por la complejidad de la nube, los contenedores y la infraestructura como código.

Fallas comunes:

- Cuentas y contraseñas por defecto que nunca se cambiaron.
- Servicios, puertos o funciones innecesarias habilitadas.
- Mensajes de error detallados expuestos al usuario final.
- Permisos abiertos en almacenamiento en la nube.
- Encabezados de seguridad ausentes o mal configurados.

Cómo prevenirlo:

- Definir un proceso de endurecimiento repetible y automatizado para todos los ambientes.
- Mantener ambientes de desarrollo, pruebas y producción configurados de forma equivalente.
- Eliminar componentes y funciones que no se usan.
- Revisar la configuración como parte del pipeline, no como tarea manual.

A03 — Software Supply Chain Failures

Categoría nueva en 2025, que reemplaza y amplía la antigua Vulnerable and Outdated Components.

Ya no se trata solo de usar una biblioteca con vulnerabilidades conocidas, sino de todo el proceso de construir, distribuir y actualizar software.

Alcance de la categoría:

- Dependencias directas y transitivas sin control de versiones.

- Componentes obtenidos de fuentes no confiables.
- Pipelines de CI/CD con permisos excesivos o sin separación de funciones.
- Repositorios, estaciones de trabajo y servidores de construcción sin endurecer.

Casos reales:

- SolarWinds (2019), donde actualizaciones firmadas del proveedor llevaron malware a unas 18,000 organizaciones.
- Log4Shell (CVE-2021-44228), una vulnerabilidad crítica en una biblioteca de registro presente en medio mundo.
- Shai-Hulud (2025), un gusano autopropagante en npm que infectó más de 500 paquetes y robó credenciales de desarrolladores.

Cómo prevenirlo:

- Mantener un SBOM (Software Bill of Materials) del proyecto.
- Fijar versiones exactas y verificar integridad con firmas o sumas de verificación.
- Monitorear bases de vulnerabilidades como CVE, NVD y OSV de forma continua.
- Obtener componentes solo de fuentes oficiales y por canales seguros.
- Usar autenticación multifactor y menor privilegio en repositorios y pipelines.

A04 — Cryptographic Failures

El foco está en la protección de los datos, en tránsito y en reposo.

Antes se llamaba Sensitive Data Exposure, nombre que describía el síntoma; el nombre actual apunta a la causa raíz.

Fallas comunes:

- Transmitir datos en texto claro, sin TLS.
- Usar algoritmos obsoletos o débiles como MD5 o SHA-1.
- Guardar contraseñas con hash simple en lugar de funciones diseñadas para contraseñas.
- Claves de cifrado versionadas en el repositorio o generadas con aleatoriedad predecible.

Cómo prevenirlo:

- Clasificar los datos y definir qué requiere cifrado según su sensibilidad.
- No almacenar datos sensibles que no sean necesarios.
- Usar TLS actualizado en todo tránsito y algoritmos vigentes en reposo.
- Almacenar contraseñas con Argon2, bcrypt o scrypt, con salt por usuario.
- Gestionar las claves en un servicio dedicado, con rotación definida.

A05 — Injection

Ocurre cuando datos no confiables se envían a un intérprete como parte de un comando o consulta.

Incluye inyección SQL, NoSQL, de comandos del sistema operativo, LDAP y también Cross-Site Scripting.

Bajó del puesto 3 al 5, no porque el riesgo desaparezca, sino porque los frameworks modernos ya traen defensas por defecto.

Fallas comunes:

- Construir consultas concatenando cadenas con entrada del usuario.

- Pasar parámetros directamente a comandos del sistema operativo.
- Insertar entrada del usuario en el HTML de la página sin escapar.

Cómo prevenirlo:

- Usar consultas parametrizadas o un ORM que las genere.
- Validar la entrada en el servidor con listas de permitidos.
- Escapar la salida según el contexto donde se inserta.
- Limitar los registros devueltos por una consulta para acotar la fuga en caso de éxito del ataque.

A06 — Insecure Design

Un diseño inseguro no se arregla con una implementación perfecta.

La distinción es clave: un defecto de implementación es un error al escribir el código; un defecto de diseño es la ausencia de un control que nunca se pensó.

Fallas comunes:

- Flujos de recuperación de contraseña basados en preguntas de seguridad adivinables.
- Ausencia de límites de negocio, como permitir comprar mil boletos en una sola transacción automatizada.
- No considerar escenarios de abuso al definir los requerimientos.

Cómo prevenirlo:

- Hacer modelado de amenazas sobre los flujos críticos del sistema.
- Escribir casos de abuso junto a los casos de uso.

- Usar patrones de diseño seguros y bibliotecas de componentes ya endurecidos.
- Separar por capas y por inquilinos según el nivel de exposición de cada parte.

A07 — Authentication Failures

Confirmar la identidad del usuario y sostener esa identidad durante la sesión.

En 2025 cambió de nombre, de Identification and Authentication Failures a Authentication Failures.

Fallas comunes:

- Permitir contraseñas débiles o filtradas en brechas conocidas.
- No limitar los intentos de inicio de sesión, habilitando fuerza bruta y credential stuffing.
- Identificadores de sesión expuestos en la URL o que no se invalidan al cerrar sesión.
- Procesos de recuperación de cuenta más débiles que el inicio de sesión mismo.

Cómo prevenirlo:

- Implementar autenticación multifactor donde el riesgo lo justifique.
- Verificar las contraseñas contra listas de credenciales filtradas.
- Aplicar límites de intentos y retardos progresivos, registrando los fallos.
- Generar identificadores de sesión aleatorios del lado del servidor, con expiración e invalidación explícita.

A08 — Software or Data Integrity Failures

Se trata de confiar en código o datos sin verificar su origen ni su integridad.

Fallas comunes:

- Actualizaciones automáticas que se aplican sin verificar la firma del paquete.
- Deserializar objetos provenientes de una fuente no confiable.
- Incluir bibliotecas desde CDN sin control de integridad del recurso.
- Pipelines que despliegan artefactos sin validar que provengan de la construcción esperada.

Cómo prevenirlo:

- Verificar firmas digitales en actualizaciones y artefactos.
- Evitar la deserialización de datos no confiables o restringirla a tipos conocidos.
- Usar Subresource Integrity al cargar recursos externos.
- Proteger el pipeline con revisión de cambios y segregación de funciones.

A09 — Security Logging and Alerting Failures

Sin registro y alerta no hay detección, y sin detección un incidente puede durar meses sin que nadie se entere.

En 2025 el nombre pasó de monitoring a alerting, para enfatizar que registrar no basta si nadie reacciona.

Fallas comunes:

- Eventos de autenticación y de control de acceso que no se registran.
- Registros que solo quedan en el servidor local y se pierden con él.
- Alertas que nadie revisa o umbrales mal definidos.

- Registros que guardan datos sensibles en texto claro.

Cómo prevenirlo:

- Registrar inicios de sesión, fallos de acceso y transacciones de alto valor con contexto suficiente.
- Centralizar los registros y protegerlos contra alteración.
- Definir alertas accionables y quién responde a cada una.
- Tener un plan de respuesta a incidentes y ejercitarlo.

A10 — Mishandling of Exceptional Conditions

Categoría nueva en 2025, agrupa 24 CWE sobre el manejo incorrecto de errores y condiciones inesperadas.

La idea central es que el comportamiento del sistema cuando algo sale mal también es una decisión de diseño.

Fallas comunes:

- Fallar de forma abierta, donde un error en la verificación termina concediendo el acceso.
- Devolver trazas de la excepción al usuario, revelando estructura interna del sistema.
- Capturar excepciones de forma genérica y continuar como si nada hubiera pasado.
- Dejar el sistema en estado inconsistente tras un error a medio camino.

Cómo prevenirlo:

- Fallar de forma cerrada, denegando por defecto cuando la verificación no se completa.
- Devolver mensajes de error genéricos al usuario y el detalle solo al registro.
- Manejar cada excepción de forma explícita y probar las rutas de error.
- Usar transacciones para que una operación fallida no deje datos a medias.

Qué cambió respecto a 2021

- **Dos categorías nuevas:** A03 Software Supply Chain Failures y A10 Mishandling of Exceptional Conditions.
- **Tres renombramientos:** A07 se simplificó a Authentication Failures, A09 pasó de monitoring a alerting, y la antigua categoría de componentes se amplió a toda la cadena de suministro.
- **Una fusión:** SSRF dejó de ser categoría propia y quedó dentro de A01.
- **Reordenamientos:** Security Misconfiguration subió del 5 al 2, Cryptographic Failures bajó del 2 al 4, Injection del 3 al 5 e Insecure Design del 4 al 6.

La tendencia de fondo es clara: el riesgo se movió del código aislado hacia el proceso, la configuración y las dependencias.

Prácticas para el equipo de desarrollo

- **Modelado de amenazas** en el diseño de cada funcionalidad crítica.
- **SAST** (análisis estático) sobre el código propio en cada integración.
- **SCA** (análisis de composición) sobre las dependencias, con SBOM actualizado.
- **DAST** (análisis dinámico) sobre la aplicación desplegada, con herramientas como OWASP ZAP.

- **Revisión de código** con una lista de verificación de seguridad, no solo de estilo.
- **Pruebas de regresión de seguridad** para que un defecto corregido no regrese.

Límites del Top 10

El Top 10 es un punto de partida, no un programa completo de seguridad.

- Cubre riesgos frecuentes, no todos los riesgos del sistema.
- No reemplaza al modelado de amenazas propio de la aplicación.
- Para verificación formal existe el OWASP ASVS, que define niveles y requisitos comprobables.

Una aplicación puede pasar el Top 10 y seguir siendo insegura frente a las amenazas específicas de su dominio.

Recursos

- OWASP Top 10:2025 — <https://top10.owasp.org/2025/>
- OWASP Cheat Sheet Series — <https://cheatsheetseries.owasp.org/>
- OWASP ASVS — <https://owasp.org/www-project-application-security-verification-standard/>
- OWASP ZAP — <https://www.zaproxy.org/>