OWASP

Ciberseguridad

ÁLVARO ALLÉN PERLINES

Contenido

Introducción: ¿qué es OWASP?

OWASP (Open Web Aplication Security Proyect) es una organización internacional sin ánimo de lucro dedicada a mejorar la seguridad del software. Su objetivo principal es concienciar a desarrolladores, empresas y profesionales de la informática sobre los riesgos de seguridad más comunes en aplicaciones web, así como proporcionar documentación, herramientas y buenas prácticas para prevenirlos. Uno de sus proyectos más conocidos es el **OWASP TOP 10,** una lista que identifica las diez vulnerabilidades de seguridad más críticas basándose en datos reales de ataques y fallos detectados en aplicaciones web a nivel mundial. Esta clasificación es “en tiempo real”, es decir, cada cierto tiempo cambia según el momento tecnológico. Por ejemplo, su última actualización es de 2021 y desde entonces no se ha modificado.

Top 10 Vulnerabilidades OWASP

Control de acceso roto (Broken Access Control)

El control de acceso hace cumplir las políticas de forma que los usuarios no puedan actuar fuera de los permisos que les corresponden. Los fallos suelen provocar la divulgación no autorizada de información, la modificación o destrucción de todos los datos, o la ejecución de funciones de negocio fuera del límite del usuario. Las vulnerabilidades más comunes son:

- Violación del principio de mínimo privilegio, comúnmente llamado “denegar por defecto”, donde el acceso solo debería concederse para capacidades, roles o usuarios concretos.
- Elusión de las comprobaciones de control de acceso modificando la URL, el estado interno de la aplicación o la página HTML, o mediante el uso de herramientas de ataque que modifican las peticiones a la API.
- Permitir ver o editar la cuenta de otra persona proporcionando su identificador único.
- Una API con controles de acceso ausentes para las operaciones POST, PUT y DELETE.
- Elevación de privilegios: actuar como un usuario sin haber iniciado sesión o conseguir privilegios superiores a tu rol.

Como prevenirlo

El control de acceso solo es efectivo cuando se implementan en código de servidor de confianza o en APIs serverless, donde el atacante no puede modificar la comprobación de acceso ni los metadatos.

- Salvo para recursos públicos, denegar por defecto.
- Implementar los mecanismos de control de acceso una sola vez y reutilizarlos en toda la aplicación.
- Los controles de acceso deben hacer cumplir la propiedad de los registros.
- Registrar los fallos de control de acceso.

Ejemplos de ataques.

La aplicación utiliza datos no verificados en una llamada SQL que accede a información de cuentas.

“

Pstmt.setString(1, request.getParameter(“acct”);

ResultSet results = pstmt.executeQuery();

”

Un atacante puede simplemente modificar el parámetro acct del navegador para enviar cualquier número de cuenta deseado. Si no se verifica correctamente, el atacante puede acceder a la cuenta de cualquier usuario.

“

https://example.com/app/accountInfo?acct=notmyacct

”

Bibliografía:

- A01 Broken Access Control - OWASP Top 10:2025

Fallos criptográficos (Cryptographic Failures)

La mala configuración de seguridad se produce cuando un sistema, una aplicación o un servicio en la nube se configura incorrectamente desde el punto de vista de la seguridad, creando vulnerabilidades.

La aplicación puede ser vulnerable si:

- Falta un endurecimiento de seguridad adecuado en cualquier parte de la pila de la aplicación.
- Hay funcionalidades innecesarias habilitadas o instaladas.
- Las cuentas por defecto y sus contraseñas siguen habilitadas y no se han modificado.
- Falta una configuración centralizada para interceptar mensajes de error excesivos.
- A la hora de actualizar, configurar incorrectamente o deshabilitar funcionalidades de seguridad.
- Priorizar compatibilidad hacía atrás (retrocompatibilidad), respecto a seguridad.

Como prevenirlo

Deben implementarse procesos de instalación segura, que incluyan:

- Un proceso de “hardening” repetible que permita el despliegue rápido y sencillo de otro entorno correctamente protegido.
- Una plataforma mínima, sin funcionalidades, componentes, documentación o ejemplos innecesarios.
- Una arquitectura de aplicación segmentada que proporcione una separación efectiva y segura entre componentes o inquilinos, mediante segmentación.
- Envío de directivas de seguridad a los clientes.

Ejemplos de escenarios de ataque

El servidor de aplicaciones incluye aplicaciones de ejemplo que no se han eliminado del servidor de producción. Estas aplicaciones de ejemplo tienen fallos de seguridad conocidos que los atacantes aprovechan para comprometer el servidor. Supongamos que una de estas aplicaciones es la consola de administración y que las cuentas no se han cambiado. En este caso, el atacante inicia sesión con la contraseña por defecto.

Bibliografía:

- A02 Security Misconfiguration - OWASP Top 10:2025

Fallos en la cadena de suministro de software

Son interrupciones u otros compromisos en el proceso de construcción, distribución o actualización del software. Suelen estar causados por vulnerabilidades o cambios maliciosos en código de terceros, herramientas u otras dependencias de las que depende el sistema.

Algunas situaciones de vulnerabilidad son:

- No realizar un seguimiento cuidadoso de las versiones de todos los componentes que utilizas. Esto incluye los componentes que usas directamente y las dependencias anidadas.
- El software es vulnerable, no está soportado o está desactualizado.
- No realizas escaneos de vulnerabilidades de forma regular.
- No dispones de un proceso de gestión de cambios ni de un sistema de seguimiento de cambios dentro de tu cadena de suministro.
- No has reforzado todas las partes de tu cadena de suministro.
- Se utilizan componentes de fuentes no confiables.

Como prevenirlo

Debe existir un proceso de gestión de parches para:

- Generar y gestionar de forma centralizada el SBOM de todo el software.
- Realizar el seguimiento no solo de las dependencias directas, sino también de sus dependencias transitivas.
- Reducir la superficie de ataque eliminando dependencias no utilizadas, funcionalidades innecesarias, componentes, archivos y documentación.
- Inventariar de forma continua las versiones de los componentes tanto del lado cliente como del servidor.
- Monitorizar de forma ocntinua fuentes como CVE, NVD y OSV para detectar vulnerabilidades en los componentes utilizados.
- Obtener componentes únicamente de fuentes oficiales mediante enlaces seguros.

- Elegir deliberadamente la versión de cada dependencia y actualizar solo cuando sea necesario.

Ejemplos de ataques

Un proveedor de confianza se ve comprometido con malware, lo que provoca que tus sistemas se comprometan cuando realizas una actualización. El ejemplo más conocido es:

El compromiso de SolarWinds en 2019, que provocó que unas 18000 organizaciones se vieran afectadas.

Bibliografía:

- A03 Software Supply Chain Failures - OWASP Top 10:2025
- Información sobre el ejemplo expuesto: Ciberataque a SolarWinds

Fallos en criptografía

En términos generales, todos los datos en tránsito deben cifrarse en la capa de transporte. Obstáculos anteriores como el rendimiento de la CPU y la gestión de claves privadas/certificados hoy en día están resueltos gracias a CPUs con instrucciones diseñadas para acelerar el cifrado y a la simplificación de la gestión de claves y certificados mediante servicios como Let´s Encrypt.

Es importante determinar qué datos necesitan cifrado en reposo y qué datos requieren un cifrado adicional en tránsito. Por ejemplo, las contraseñas necesitan de este cifrado adicional. Por ello, deben plantearse las siguientes cuestiones:

- ¿Se utilizan algoritmos o protocolos criptográficos antiguos o débiles?
- ¿Se utilizan claves criptográficas por defecto, se generan claves débiles, se reutilizan claves o falta una gestión y rotación adecuadas de claves?
- ¿Las claves criptográficas se incluyen en repositorios de código fuente?
- ¿No se fuerza el cifrado, por ejemplo, falta directivas o cabeceras de seguridad HTTP?
- ¿Se utilizan funciones hash obsoletas como MD5 o SHA1?

Como prevenirlo

Como mínimo, haz lo siguiente y consulta las referencias:

- Clasifica y etiqueta los datos que la aplicación procesa, almacena o transmite.
- Almacena las claves más sensibles en un HSM basado en hardware o en la nube.
- Utiliza implementaciones de algoritmos criptográficos ampliamente confiables siempre que sea posible.
- No almacenes datos sensibles innecesariamente.

- Asegúrate de cifrar todos los datos sensibles en reposo.
- Utiliza siempre cifrado autenticado en lugar de solo cifrado.

Ejemplos de ataques

Un sitio no utiliza ni fuerza TLS en todas sus páginas o admite cifrado débil. Un atacante monitoriza el tráfico de red, degrada las conexiones de HTTPS a HTTP, intercepta las peticiones y roba la cookie de sesión del usuario. A continuación, el atacante reutiliza esa cookie y secuestra la sesión del usuario accediendo o modificando sus datos privados. Alternativamente, podría modificar los datos transportados, por ejemplo el destinatario de una transferencia de dinero.

Bibliografía:

- A04 Cryptographic Failures - OWASP Top 10:2025

Inyección

Una vulnerabilidad de inyección es un fallo de la aplicación que permite que datos de entrada no confiables proporcionados por el usuario se envíen a un intérprete y provoquen que el intérprete ejecute partes de esa entrada como si fueran comandos.

Una aplicación es vulnerable a ataques de inyección cuando:

- Los datos proporcionados por el usuario no se validan, filtran ni sanean.
- Se utilizan consultas dinámicas o llamadas no parametrizadas sin un escapado adecuado.
- Se usan datos no saneados en parámetros de búsqueda.
- Datos potencialmente maliciosos se usan o concatenan directamente.

Como prevenirlo

La mejor forma de prevenir la inyección consiste en mantener los datos separados de los comandos y las consultas:

- La opción preferida es utilizar una API segura que evite por completo el uso directo del intérprete, que proporcione una interfaz parametrizada o que migre herramientas de mapeo.
- Cuando no sea posible separar los datos de los comandos, se pueden reducir los riesgos utilizando las siguientes técnicas.
    - Usar validación positiva de entradas en el servidor. No es una defensa completa ya que muchas aplicaciones requieren de caracteres especiales.
    - Para las consultas dinámicas residuales, escapar los caracteres especiales utilizando la sintaxis específica del intérprete.

Ejemplos de ataques

Una aplicación utiliza datos no confiables para construir la siguiente llamada SQL vulnerable:

“

String query = “SELECT * FROM accounts WHERE custID=’” + request.getParameter(“id”) + “’”;

”

Un atacante modifica el valor del parámetro id en el navegador para enviar ‘ OR ‘1’ =’1.

Esto modifica el significado de la consulta y hace que devuelva todos los registros de la tabla “accounts”. Ataques más peligrosos podrían modificar o eliminar datos o incluso invocar procedimientos almacenados.

Bibliografía:

- A04 Cryptographic Failures - OWASP Top 10:2025

Diseño inseguro

El diseño inseguro es una categoría que representa diferentes debilidades, expresadas como “controles de diseño inexistentes o ineficaces”. Existe una gran diferencia entre diseño inseguro e implementación insegura: tienen causas raíz diferentes, ocurren en momentos distintos del proceso de desarrollo y requieren medidas correctivas diferentes.

Un diseño seguro puede seguir teniendo defectos de implementación que deriven en vulnerabilidades explotables. Sin embargo, un diseño inseguro no puede arreglarse con una implementación perfecta, ya que los controles de seguridad necesarios nunca se crearon para defenderse de ataques concretos. Uno de los factores que contribuyen al diseño inseguro es la falta de un perfilado del riesgo de negocio inherente al software o sistema que se está desarrollando.

Tres partes clave para disponer de un diseño seguro son:

1. Recopilación de requisitos y gestión de recursos.
2. Creación de un diseño seguro.
3. Disponer de un ciclo de vida de desarrollo seguro.

Como prevenirlo

- Establecer y utilizar un ciclo de vida de desarrollo seguro con profesionales que ayuden a evaluar y diseñar controles de seguridad y privacidad.
- Establecer y utilizar una librería de patrones de diseño seguro o componentes paved-road.

- Utilizar modelado de amenazas para partes críticas de la aplicación como autenticación, control de acceso, lógica de negocio y flujos clave.
- Usar el modelado de amenazas como herramienta educativa para generar una mentalidad de seguridad.
- Integrar lenguaje y controles de seguridad en las historias de usuario.

Ejemplos de ataques

Un flujo de recuperación de credenciales incluye “preguntas y respuestas”, lo cual está prohibido por NIST 800-63b, OWASP ASVS y el OWASP Top 10. Las preguntas y respuestas no pueden considerarse una prueba fiable de identidad, ya que más de una persona puede conocerlas. Esta funcionalidad debería eliminarse y sustituirse por un diseño más seguro.

Bibliografía:

- A06 Insecure Design - OWASP Top 10:2025

Fallos de identificación y autenticación

Esta vulnerabilidad se produce cuando un atacante es capaz de engañar a un sistema para que reconozca como legítimo a un usuario inválido o incorrecto. Pueden existir debilidades de autenticación si la aplicación:

- Permite ataques automatizados en los que el atacante dispone de una lista filtrada de combinaciones válidas de nombres de usuario y contraseñas.
- Permite ataques de fuerza bruta u otros ataques automatizados.
- Permite contraseñas por defecto, débiles o ampliamente conocidas.
- Permite a los nuevos usuarios crear nuevas cuentas utilizando credenciales que ya han sido comprometidas.
- Permite el uso de procesos débiles o ineficaces de recuperación de credenciales u “olvide mi contraseña”.
- Carece de autenticación multifactor o ésta es ineficaz.
- Reutiliza el identificador de sesión tras un inicio exitoso.

Como prevenirlo

- Siempre que sea posible, implementar o forzar el uso de autenticación multifactor para prevenir ataques automatizados.
- Cuando sea posible, fomentar y facilitar el uso de gestores de contraseñas, para ayudar a los usuarios a tomar mejores decisiones.
- No distribuir ni desplegar sistemas con credenciales por defecto.
- No obligar a las personas a rotar contraseñas salvo que se sospeche una brecha de seguridad.

- Limitar o retrasar progresivamente los intentos de inicio de sesión fallidos, teniendo cuidado de no provocar escenarios de denegación de servicio.

Ejemplos de ataques

Recientemente se ha observado que los atacantes “incrementan” o ajustan las contraseñas basándose en comportamientos humanos habituales. Este ajuste de intentos de contraseña se conoce como ataque híbrido, y puede ser incluso más eficaz que la versión tradicional.

Bibliografía:

- A07 Authentication Failures - OWASP Top 10:2025

Fallos en la Integridad de Software y Datos

Los fallos de integridad del software y de los datos se refieren a código e infraestructuras que no protegen frente a que código o datos inválidos o no confiables sean tratados como confiables y válidos.

Una canalización CI/CD insegura que no consuma ni proporcione comprobaciones de integridad del software puede introducir la posibilidad de accesos no autorizados, código inseguro o malicioso, o la compromisión del sistema.

Además, muchas aplicaciones actuales incluyen funcionalidades de actualización automática, en las que las actualizaciones se descargan sin una verificación suficiente de la integridad y se aplican a una aplicación previamente confiable. Los atacantes podrían subir sus propias actualizaciones para que se distribuyan y ejecuten en todas las instalaciones.

Como prevenirlo

- Utilizar firmas digitales u otros mecanismos similares para verificar que el software o los datos proceden de la fuente esperada y no han sido alterados.
- Asegurar que las librerías y dependencias solo se obtienen de repositorios confiables.
- Garantizar que exista un proceso de revisión para los cambios de código y configuración.
- Asegurar que la canalización CI/CD cuente con una correcta segregación, configuración y control de accesos para proteger la integridad del código.
- Asegurar que no se reciban datos serializados no firmados o no cifrados desde clientes no confiables para ser utilizados posteriormente sin algún tipo de comprobación de integridad o firma digital.

Ejemplos de ataques

Una empresa utiliza un proveedor externo para ofrecer funcionalidades de soporte. Esto implica que todas las cookies se enviarán ahora al proveedor de soporte. Cualquier persona con acceso a la infraestructura del proveedor podría robar las cookies de todos los usuarios que hayan visitado la página y llevar a cabo un ataque de secuestro de sesión.

Bibliografía:

- A08 Software or Data Integrity Failures - OWASP Top 10:2025

Fallos en Registro y Monitorización

Sin un registro o monitorización adecuados, los ataques y brechas no pueden detectarse, y sin alertas resulta muy difícil responder de forma rápida y eficaz durante un incidente de seguridad. La falta de registros, monitorización continua, detección y alertas para iniciar respuestas activas se produce cuando:

- Los eventos auditables, como inicios de sesión, intentos fallidos de inicio de sesión y transacciones de alto valor, no se registran o se registran de forma inconscientes.
- Las advertencias y errores no generan registros.
- La integridad de los registros no está debidamente protegida frente a manipulaciones.
- Los registros de aplicaciones y APIs no se monitorizan para detectar actividades sospechosas.
- Los registros solo se almacenan localmente y no se respaldan correctamente.
- La aplicación no puede detectar, escalar o alertar sobre ataques activos en tiempo real o casi en tiempo real.

Como prevenirlo

Los desarrolladores deberían implementar algunos a todos los siguientes controles, en función del riesgo de la aplicación:

- Asegurar que todos los fallos de inicio de sesión, control de acceso y validación de entradas en el servidor se registran con suficiente contexto del usuario.
- Garantizar que cada parte de la aplicación que contenga un control de seguridad quede registrada, tanto si el control tiene éxito como si falla.
- Asegurar que los registros se generan en un formato que las soluciones de gestión de logs puedan consumir fácilmente.
- Análisis de comportamiento y el uso de IA pueden ser técnicas opcionales adicionales para mantener bajos los índices de falsos positivos.

Ejemplos de ataques

El operador del sitio web de un proveedor de planes de salud infantiles no pudo detectar una brecha debido a la falta de monitorización y registros. Una parte externa informó al proveedor de que un atacante había accedido y modificado miles de registros sanitarios sensibles de más de 3.5 millones de niños. Una revisión posterior al incidente reveló que los desarrolladores del sitio web no habían corregido vulnerabilidades significativas.

Bibliografía:

- A09 Security Logging and Alerting Failures - OWASP Top 10:2025

La gestión incorrecta de condiciones excepcionales

La gestión incorrecta de condiciones excepcionales en el software ocurre cuando los programas no previenen, detectan o responden adecuadamente a situaciones inusuales e impredecibles, lo que provoca caídas, comportamientos inesperados y, en ocasiones, vulnerabilidades.

Las condiciones excepcionales pueden deberse a validaciones de entrada inexistentes, deficientes o incompletas; a un manejo de errores tardío y de alto nivel en lugar de hacerlo en las funciones donde se producen; o a estados inesperados del entorno como problemas de memoria, privilegios o red. También pueden originarse por una gestión inconscientes de excepciones o por excepciones que no se manejan en absoluto, permitiendo que el sistema entre en un estado desconocido e impredecible.

Muchas vulnerabilidades de seguridad pueden surgir al gestionar incorrectamente condiciones excepcionales, como fallos de lógica, desbordamientos, condiciones de carrera, etc… Este tipo de vulnerabilidades pueden afectar negativamente a la confidencialidad, disponibilidad y/o integridad de un sistema o de sus datos.

Como prevenirlo

- Capturar y gestionar condiciones excepcionales garantizar que la infraestructura subyacente del programa no tenga que lidiar con situaciones impredecibles. Si una transacción está en curso, es fundamental revertirla por completo y comenzar de nuevo.
- Siempre que sea posible, se deben aplicar límites como “rate limiting”, cuotas de recursos y mecanismos de “throttling” para prevenir condiciones excepcionales desde el principio.
- Conviene considerar si errores idénticos o repetidos deberían registrarse únicamente como estadísticas que indiquen cuántas veces han ocurrido y en qué periodo.
- Se debe aplicar una validación estricta de entradas, junto con un manejo centralizado de errores, registros, monitorización y alertas, y un gestor global de excepciones.

- Toda la organización debería gestionar las condiciones excepcionales de la misma manera, ya que esto facilita la revisión y auditoría del código en este control de seguridad tan importante.

Ejemplos de ataques

El agotamiento de recursos debido a una mala gestión de condiciones excepcionales puede producirse si la aplicación captura excepciones durante la subida de archivos, pero no libera correctamente los recursos después. Cada nueva excepción deja recursos bloqueados o no disponibles, hasta que se agotan por completo.

Bibliografía:

- A10 Mishandling of Exceptional Conditions - OWASP Top 10:2025