Entendiendo los cambios entre OWASP Top 10 2017 y 2021

Juan Felipe Rojas Cendales
Departamento de Ingeniería de Sistemas e Industrial, Universidad Nacional de Colombia
Bogotá, Colombia
jrojasce@unal.edu.co

***Abstract*— Este informe realiza un análisis comparativo entre las ediciones 2017 y 2021 del OWASP Top 10, un documento ampliamente reconocido por la comunidad de la ciberseguridad que identifica las principales vulnerabilidades en aplicaciones web. El objetivo principal de este informe es examinar la evolución y posibles cambios de las distintas vulnerabilidades, explicar a profundidad cada una de estas, destacar tendencias en el sector de la ciberseguridad y exponer las recomendaciones en medidas de prevención. Finalmente, se concluye la importancia de la seguridad en todas las etapas de desarrollo y el constante ciclo de estudio, adaptación y fortalecimiento de la ciberseguridad ante el surgimiento de nuevas tecnologías y el constante avance en el sector.**

I. INTRODUCCIÓN

El OWASP Top 10 es un proyecto comunitario que identifica las principales vulnerabilidades en las aplicaciones web, proporcionando orientación y estándares valiosos para desarrolladores, diseñadores, managers, arquitectos y expertos en seguridad. Este informe realiza un análisis comparativo entre las ediciones de 2017 y 2021 con el objetivo de examinar la evolución y cambios de las vulnerabilidades de seguridad en aplicaciones web durante este período de 5 años.

Cada edición establece un marco sólido y variado al identificar las principales vulnerabilidades que podrían explotarse para comprometer la seguridad de las aplicaciones web. Sin embargo, el contraste de dos ediciones refleja cambios en el panorama de la ciberseguridad, presentando una nueva lista de las principales vulnerabilidades ligadas al cambio y evolución en el sector tecnológico.

El informe revisará ambos OWASP Top 10 por separado, presentando cada vulnerabilidad descrita en estos para posteriormente enumerar los distintos cambios y analizar y concluir sobre estos.

II. OWASP 2017

El OWASP Top 10 2017 se basa en más de 40 contribuciones de datos de empresas especializadas en seguridad de aplicaciones y una encuesta de la industria completada por más de 500 individuos. Estos datos abarcan vulnerabilidades recopiladas de cientos de organizaciones y más de 100,000 aplicaciones y APIs del mundo real.

La lista de vulnerabilidades se presenta a continuación.

A. Injection

La inyección permite a los atacantes ejecutar comandos en el lenguaje de consulta (SQL, NoSQL, ORM, etc) no autorizados dentro de una aplicación web. Los atacantes pueden aprovechar esta vulnerabilidad para leer, crear, modificar o eliminar datos almacenados en la base de datos. Este tipo de ataque se da cuando la aplicación no valida correctamente las entradas de cara al usuario como formularios, parámetros URL u otros y les permite insertar consultas maliciosas.

Para prevenir la inyección, es fundamental utilizar APIS seguras, consultas parametrizadas, ORM's, procedimientos almacenados o declaraciones separadas para evitar la concatenación de cadenas directamente en la consulta. Además, es importante validar y sanitizar todas las entradas del usuario antes de enviarlas a la base de datos junto con limitaciones como LIMIT en la consulta de datos. Finalmente, se recomienda el uso de funciones de escape de caracteres y aplicar el principio de mínimo privilegio en los diversos componentes del sistema.

Supongamos que una aplicación web tiene un formulario de inicio de sesión. Si el campo de contraseña no se valida adecuadamente y un atacante ingresa una cadena cualquiera acompañada de OR '1'='1’, la consulta SQL resultante podría acceder a la cuenta de cualquier usuario sin necesidad de la contraseña.

B. Broken Authentication

La autenticación rota se refiere a las vulnerabilidades relacionadas con el manejo incorrecto de la autenticación y la gestión de sesiones. Esto puede permitir a los atacantes comprometer cuentas de usuario, realizar ataques de fuerza bruta y reutilizar credenciales robadas.

Para prevenir la autenticación rota, es fundamental implementar autenticación multifactorial siempre que sea posible, evitar el uso de credenciales por defecto, validar y rechazar el uso de contraseñas débiles siguiendo las pautas modernas establecidas como NIST. Además, es esencial reforzar los flujos de registro, inicio de sesión y recuperación de credenciales, limitar o retrasar los intentos de inicio de sesión fallidos, y utilizar un administrador de sesiones seguro en el lado del servidor.

Supongamos que una aplicación web permite a un atacante realizar un número ilimitado de intentos de inicio de sesión sin limitaciones, lo cual facilita un ataque de fuerza bruta para adivinar las credenciales de un usuario.

C. Sensitive Data Exposure

La exposición de datos sensibles puede comprometer la confidencialidad de la información almacenada o transmitida por los distintos flujos de una aplicación web.

Para prevenir la exposición de datos sensibles, es fundamental categorizar los datos sensibles, almacenar sólo los datos

necesarios, cifrar todos los datos sensibles utilizando algoritmos y claves fuertes, manteniendo actualizados los estándares de cifrado. Además, se debe garantizar la encriptación de todos los datos en tránsito utilizando protocolos seguros como TLS con cifrados de secreto perfecto hacia adelante (PFS) y parámetros seguros. Es importante deshabilitar el almacenamiento en caché para las respuestas que contienen datos sensibles y almacenar contraseñas utilizando funciones de hash fuertes.

Supongamos que una aplicación web almacena las contraseñas de los usuarios en una base de datos sin cifrar. Un atacante logra acceder a esta base de datos y robar las contraseñas de los usuarios para uso inmediato.

D. XML External Entities

Las Entidades Externas XML (XXE) permiten a un atacante acceder, leer o incluso modificar datos sensibles al procesar documentos XML maliciosos. Esta vulnerabilidad se produce cuando una aplicación web procesa entradas XML no confiables que contienen referencias a entidades externas, como archivos locales o recursos remotos, lo que puede llevar a una ejecución remota de código, fugas de información o denegación de servicio.

Para prevenir las entidades externas XML, es fundamental que los desarrolladores reciban una capacitación adecuada, utilizar formatos de datos menos complejos como JSON si es posible, aplicar parches o actualizaciones a todos los procesadores y bibliotecas XML, deshabilitar el procesamiento de entidades externas XML, validar entradas de cara al usuario, utilizar herramientas de análisis estático de seguridad (SAST) para detectar XXE en el código fuente.

Supongamos que una aplicación web permite a los usuarios cargar archivos XML para procesarlos y generar informes. Un atacante podría cargar un archivo XML malicioso que contiene una entidad externa que hace referencia a un archivo local en el servidor y puede acceder al contenido del mismo.

E. Broken Access Control

El Control de Acceso Roto ocurre cuando las restricciones de acceso a recursos no se aplican correctamente, permitiendo a un atacante acceder a información o funcionalidades protegidas. Esto puede resultar en la ejecución de acciones no autorizadas en la aplicación y captura de información protegida. Es efectivo cuando se aplica en el código del servidor.

Para prevenir el Control de Acceso Roto, es fundamental que se niegue el acceso por defecto a todo excepto los recursos públicos, implementar mecanismos de control de acceso una vez y reutilizarlos, hacer que estos mecanismos hagan cumplir la propiedad sobre un recurso, deshabilitar la lista de directorios del servidor web y asegurarse de que los metadatos de los archivos y los archivos de copia de seguridad no estén presentes dentro del root web, registrar los fallos de control de acceso y alertar, limitar tasa de acceso al API, invalidar tokens JWT después de cerrar sesión, implementar pruebas de acceso a funciones.

Supongamos que una aplicación web tiene una funcionalidad para que los usuarios puedan modificar su perfil, pero no se implementa correctamente el control de acceso. Un atacante podría manipular la solicitud HTTP para cambiar el perfil de otro usuario sin autenticarse adecuadamente. Esto podría llevar a la modificación de información personal de otros usuarios y comprometer la integridad de los datos.

F. Security Misconfiguration

La configuración de seguridad incorrecta ocurre cuando los ajustes de seguridad de una aplicación web no están correctamente configurados, lo que puede dejarla vulnerable a diversos tipos de ataques.

Para prevenir la configuración de seguridad incorrecta, es fundamental configurar los distintos entornos de manera idéntica, establecer un proceso repetible que permita desplegar rápidamente otro entorno, revisar y actualizar las configuraciones de acuerdo a los parches, implementar una arquitectura de aplicación segmentada y pruebas automatizadas para revisar las configuraciones.

Supongamos que una aplicación web tiene permisos de lectura y escritura excesivos y que no siguen el principio de privilegio mínimo en archivos del sistema (configuración incorrecta). Un atacante accede al servidor y modifica los archivos.

G. Cross-Site Scripting (XSS)

El Cross-Site Scripting (XSS) permite a los atacantes inyectar scripts maliciosos en páginas web vistas por otros usuarios. Estos scripts pueden ser utilizados para robar información, redirección o realizar acciones.

Para prevenir el Cross-Site Scripting, es fundamental el uso de frameworks que automáticamente escapen el XSS, descartar datos no confiables de las solicitudes HTTP, aplicar codificación sensible al contexto al modificar el documento del navegador como escapar caracteres especiales como <> (HTML), habilitar una política de seguridad de contenido (CSP).

Supongamos que una aplicación web cuenta con una sección de comentarios. Si un atacante inyecta un script malicioso en un comentario, este script podría ejecutarse en el navegador de otros usuarios que vean el comentario.

H. Insecure Deserialization

La Deserialización Insegura ocurre cuando los datos serializados recibidos de fuentes no confiables son deserializados por una aplicación. Esto puede resultar en la ejecución de código malicioso o la escalada de privilegios, aplicar restricciones de tipado y acceso, ejecutar el código de deserialización en entornos de bajo privilegio, registro y monitoreo constante.

Para prevenir la Deserialización Insegura, es fundamental no deserializar objetos provenientes de fuentes no confiables y permitir solo datos primitivos para estas fuentes, aplicar verificaciones de integridad como firmas digitales

Supongamos que una aplicación web deserializa objetos JSON recibidos desde un formulario de cara al usuario sin aplicar ninguna validación. Un atacante envía un JSON malicioso que ejecuta código en el servidor.

I. Using Components with Known Vulnerabilities

El Uso de Componentes con Vulnerabilidades Conocidas ocurre cuando una aplicación utiliza componentes (como librerías, frameworks, o módulos completos) que contienen vulnerabilidades conocidas por la comunidad.

Para prevenir el Uso de Componentes con Vulnerabilidades Conocidas, es fundamental implementar un proceso de gestión que elimine dependencias no utilizadas, realizar un inventario continuo

de versiones, obtener componentes de fuentes oficiales, monitorear y actualizar.

Supongamos que una aplicación web utiliza una versión desactualizada de una librería de encriptado en JavaScript. El atacante puede desencriptar la información obtenida explotando la vulnerabilidad de la librería.

J. Insufficient Logging & Monitoring

El Registro y Monitoreo Insuficiente en una aplicación web puede dificultar la detección oportuna de actividades sospechosas o maliciosas y a una respuesta nula o inadecuada ante brechas de seguridad.

Para prevenir el Registro y Monitoreo Insuficiente, es fundamental registrar todos los intentos de inicio de sesión, fallas en el control de acceso e incluir contextos adecuados, generar registros en un formato que pueda ser fácilmente consumido, garantizar que las transacciones de alto valor tengan un registro de auditoría, monitorear y construir un plan de respuesta y de recuperación ante desastres.

Supongamos que una aplicación web no monitorea su base de datos. Un atacante puede correr un script accediendo por fuerza bruta y no tendríamos registros de los accesos a la base de datos.

III. OWASP 2021

El OWASP Top 10 2021 ingresó 3 nuevas categorías.

A. Insecure Design

El Diseño Inseguro ocurre cuando se presentan fallas de diseño y arquitectura, no debe entenderse como fallas de la implementación ya que las fallas de diseño tienen una raíz distinta. No es posible solucionar estas fallas con una implementación si por definición nunca se crearon o estimaron los controles de seguridad necesarios para defenderse de ataques específicos en el proceso de diseño.

Para prevenir el Diseño Inseguro, es fundamental levantar los requerimientos de negocio y técnicos del sistema a diseñar, se deben incluir los requerimientos de protección de datos y seguridad. La adopción de una cultura y metodología de diseño es esencial para evaluar constantemente las amenazas y garantizar robustez ante ataques bien conocidos. Además, se deben establecer un ciclo de desarrollo seguro, patrones de diseño seguros y librerías de componentes seguras. Integrar controles y lenguaje de seguridad en las historias de usuario, realizar pruebas unitarias y de integración para validar flujos resistentes.

Supongamos que una aplicación web utiliza un sistema de recuperación de credenciales a través de preguntas y respuestas, algo prohibido por NIST y un fallo de diseño. Un posible atacante es cualquier persona externa que pueda conocer las respuestas.

B. Software and Data Integrity Failures

Los Fallos en el Software y la Integridad de los Datos ocurren debido a la falta de protección contra violaciones de integridad. Los ataques en esta categoría pueden ocurrir cuando el código y la infraestructura confían en plugins, librerías o módulos de fuentes no confiables, o cuando se introducen actualizaciones de software sin verificar su integridad.

Para prevenir los Fallos en el Software y la Integridad de los Datos, es fundamental el uso de firmas digitales para la verificación de autenticidad. También es importante garantizar que las librerías y dependencias se descarguen desde repositorios de confianza y verificar que no contengan vulnerabilidades conocidas. Además, se debe implementar un proceso de revisión de cambios en el código y asegurarse de que los pipelines de CI/CD tengan una adecuada configuración y control de acceso para la integridad del código.

Supongamos que una aplicación web tiene instaladas decenas de dependencias con advertencias críticas de vulnerabilidades y hace caso omiso de estas advertencias. El atacante conoce estas vulnerabilidades y explota los sitios web que las usan.

C. Server-Side Request Forgery (SSRF)

La Falsificación de Peticiones del Lado del Servidor ocurre cuando una aplicación web está recuperando un recurso remoto sin validar la URL proporcionada por el usuario. Esto permite a un atacante forzar a la aplicación a enviar una solicitud manipulada a un destino inesperado, incluso cuando está protegida por un firewall, VPN u otro tipo de lista de control de acceso (ACL) de red.

Para prevenir la Falsificación de Peticiones del Lado del Servidor, es fundamental a nivel de red segmentar la funcionalidad de acceso a recursos remotos en redes separadas y aplicar políticas de firewall que denieguen todo el tráfico intranet excepto lo esencial. A nivel de aplicación, se debe sanitizar y validar todos los datos de entrada de cara al usuario, cumplir con un esquema de URL, puerto y destino en una lista de permitidos, deshabilitar las redirecciones HTTP, y revisar constantemente la URL para evitar ataques como la reasignación de DNS y las condiciones de carrera "tiempo de comprobación, tiempo de uso" (TOCTOU).

Supongamos que el servidor de aplicación web no protege su red contra SSRF. El atacante puede conocer la red interna y determinar si los puertos están abiertos o cerrados en el servidor para iniciar un ataque.

IV. PRINCIPALES CAMBIOS

Además de la inclusión de las 3 nuevas categorías previamente presentadas es importante resaltar:

- XML External Entities, Cross-Site Scripting (XSS) y Insecure Deserialization no desaparecen, sino que, son incluidas por Security Misconfiguration, Injection y Software and Data Integrity Failures respectivamente.
- Sensitive Data Exposure cambia de nombre a Cryptographic Failures para ser más representativo a la principal causa del problema.
- Using Components with Known Vulnerabilities cambia de nombre a Vulnerable and Outdated Components para ser más conciso.
- Broken Authentication cambia de nombre a Identification and Authentication Failures.
- Insufficient Logging & Monitoring cambia de nombre a Security Logging and Monitoring Failures.
- Las vulnerabilidades más importantes en 2017 eran Injection y Broken Authentication. Sin embargo, en 2021 fueron desplazadas por Broken Access Control y Cryptographic Failures. Lo anterior puede explicarse debido a la mejora en los frameworks y las distintas herramientas de detección que automatizan las pruebas y revisión de la seguridad.
- Las malas prácticas de configuración en cuanto a la

seguridad (Security Misconfiguration) persisten y son ahora más comunes en la lista.

- La inclusión de una nueva categoría centrada en el diseño y su posición alta en el top es un llamado a tener presente la seguridad en todas las etapas de desarrollo incluyendo y enfocando a las etapas tempranas.
- Con el aumento del uso de nuevas tecnologías y enfoques tipo Server Side nacen nuevas problemáticas que se reflejan en categorías como Server-Side Request Forgery.
- La categoría más importante Broken Access Control hace un llamado a la importancia del principio de mínimo privilegio, seguir trabajando y mejorando como comunidad en herramientas de SAST y DAST y el diseño centrado en la seguridad.
- Software and Data Integrity Failures hace un llamado al uso de herramientas de escaneo de dependencias y priorizar esta tarea como parte del ciclo de desarrollo.

V. CONCLUSIONES

La evolución del OWASP Top 10 entre 2017 y 2021 refleja cambios significativos en el panorama de la ciberseguridad en las aplicaciones web. La inclusión de nuevas categorías, absorción y reestructuración de otras destacan la necesidad de abordar los nuevos y antiguos riesgos de manera más efectiva.

Se destaca la importancia de un diseño y un ciclo de vida del software que considere y maneje la seguridad desde las primeras etapas, el respaldo mediante el uso de herramientas que automaticen en la mayor parte posible las pruebas y verificaciones, la concientización sobre los diversos ataques y las vulnerabilidades existentes a la hora de cualquier desarrollo de software y el continuo riesgo en un sector que no para de evolucionar y presentar nuevas tecnologías.

REFERENCIAS

[1] OWASP Top 10 2021. 2024. https://owasp.org/Top10/

[2] OWASP Top 10 2017. 2024. https://owasp.org/www-project-top-ten/2017/Top_10