OWASP Top 10

fuente: https://owasp.org/Top10/es/A00-about-owasp/

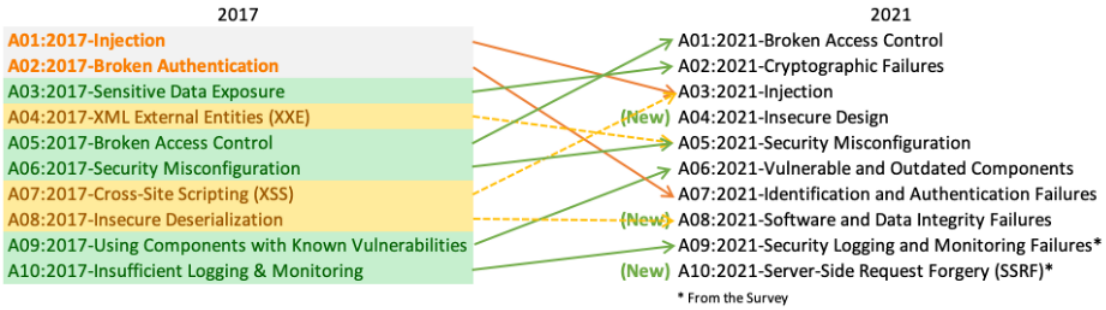
OWASP Top 10 es un documento de concientización estándar para desarrolladores y seguridad de aplicaciones web. Representa un amplio consenso sobre los riesgos de seguridad más críticos para las aplicaciones web.

Reconocido mundialmente por los desarrolladores como el primer paso hacia una codificación más segura.

Las empresas deberían adoptar este documento e iniciar el proceso para garantizar que sus aplicaciones web minimicen estos riesgos. Usar OWASP Top 10 es quizás el primer paso más efectivo para cambiar la cultura de desarrollo de software dentro de su organización hacia una que produzca código más seguro.

Los 10 principales riesgos de seguridad de las aplicaciones web

Hay tres categorías nuevas, cuatro categorías con cambios de nombre y alcance, y cierta consolidación en el Top 10 para 2021.



- **A01:2021-El control de acceso roto** sube desde la quinta posición; El 94% de las aplicaciones fueron probadas para detectar algún tipo de control de acceso roto. Las 34 enumeraciones de debilidades comunes (CWE) asignadas al control de acceso roto tuvieron más ocurrencias en las aplicaciones que cualquier otra categoría.
- **A02:2021: Las fallas criptográficas** suben una posición al n.° 2, anteriormente conocida como exposición de datos confidenciales, que era un síntoma general en lugar de una causa raíz. El enfoque renovado

aquí está en las fallas relacionadas con la criptografía, que a menudo conducen a la exposición de datos confidenciales o al compromiso del sistema.

- **A03:2021-La inyección** se desliza hacia la tercera posición. El 94% de las aplicaciones fueron probadas para detectar alguna forma de inyección, y los 33 CWE asignados a esta categoría ocupan el segundo lugar con mayor número de apariciones en las aplicaciones. Cross-site Scripting ahora forma parte de esta categoría en esta edición.
- **A04:2021: Diseño inseguro** es una nueva categoría para 2021, que se centra en los riesgos relacionados con fallas de diseño. Si realmente queremos "moverse hacia la izquierda" como industria, es necesario un mayor uso de modelos de amenazas, patrones y principios de diseño seguros y arquitecturas de referencia.
- **A05:2021: La configuración incorrecta de seguridad** asciende desde el puesto 6 en la edición anterior; El 90% de las aplicaciones fueron probadas para detectar algún tipo de configuración incorrecta. Con más cambios hacia software altamente configurable, no es sorprendente ver que esta categoría suba. La antigua categoría de entidades externas XML (XXE) ahora forma parte de esta categoría.
- **A06:2021: Componentes vulnerables y obsoletos** se titulaba anteriormente Uso de componentes con vulnerabilidades conocidas y ocupa el puesto número 2 en la encuesta comunitaria de los 10 principales, pero también tenía datos suficientes para llegar al Top 10 mediante el análisis de datos. Esta categoría asciende desde el puesto 9 en 2017 y es un problema conocido que nos cuesta probar y evaluar el riesgo. Es la única categoría que no tiene vulnerabilidades y exposiciones comunes (CVE) asignadas a los CWE incluidos, por lo que se tienen en cuenta en sus puntuaciones un exploit predeterminado y ponderaciones de impacto de 5,0.
- **A07:2021: Fallos de identificación y autenticación** anteriormente era Autenticación rota y se está deslizando hacia abajo desde la segunda posición, y ahora incluye CWE que están más relacionados con fallos de identificación. Esta categoría sigue siendo una parte integral del Top 10, pero la mayor disponibilidad de marcos estandarizados parece estar ayudando.
- **A08:2021: Fallos de integridad de datos y software** es una nueva categoría para 2021, que se centra en hacer suposiciones relacionadas con actualizaciones de software, datos críticos y canalizaciones de CI/CD

sin verificar la integridad. Uno de los impactos ponderados más altos de los datos de Vulnerabilidad y Exposiciones Comunes/Sistema de Puntuación de Vulnerabilidad Común (CVE/CVSS) asignados a los 10 CWE en esta categoría. La deserialización insegura de 2017 ahora forma parte de esta categoría más amplia.

- **A09:2021: Fallas de registro y monitoreo de seguridad** anteriormente era Registro y monitoreo insuficientes y se agrega de la encuesta de la industria (n.° 3), subiendo desde el n.° 10 anterior. Esta categoría se amplía para incluir más tipos de fallas, es difícil de probar y no está bien representada en los datos CVE/CVSS. Sin embargo, las fallas en esta categoría pueden afectar directamente la visibilidad, las alertas de incidentes y la análisis forense.
- A10:2021: Se agrega **la falsificación de solicitudes del lado del servidor de la encuesta de la comunidad Top 10 (n.º 1).** Los datos muestran una tasa de incidencia relativamente baja con una cobertura de pruebas superior al promedio, junto con calificaciones superiores al promedio de potencial de explotación e impacto. Esta categoría representa el escenario en el que los miembros de la comunidad de seguridad nos dicen que esto es importante, aunque no esté ilustrado en los datos en este momento.

A01:2021 – Pérdida de Control de Acceso

Resumen

Subiendo desde la quinta posición, el 94% de las aplicaciones fueron probadas para detectar algún tipo de pérdida de control de acceso con una tasa de incidencia promedio del 3,81%. Tuvo la mayor cantidad de ocurrencias en el conjunto de datos analizado con más de 318.000. Las CWE (Common Weakness Enumerations) más importantes incluidas son *CWE-200: Exposición de información sensible a un actor no autorizado*, *CWE-201: Exposición de información confidencial a través de datos enviados*, y *CWE-352: Falsificación de Petciones en Sitos Cruzados (Cross Site Request Forgery, CSRF por su siglas en inglés)*.

Descripción

El control de acceso implementa el cumplimiento de política de modo que los usuarios no pueden actuar fuera de los permisos que le fueron asignados. Las fallas generalmente conducen a la divulgación de información no autorizada, la modificación o la destrucción de todos los datos o la ejecución de una función de negocio fuera de los límites del usuario. Las vulnerabilidades comunes de control de acceso incluyen:

- Violación del principio de mínimo privilegio o denegación por defecto, según el cual el acceso sólo debe ser permitido para capacidades, roles o usuarios particulares, y no disponible para cualquier persona.
- Eludir las comprobaciones de control de acceso modificando la URL (alteración de parámetros o navegación forzada), el estado interno de la aplicación o la página HTML, o mediante el uso de una herramienta que modifique los pedidos a APIs.
- Permitir ver o editar la cuenta de otra persona, con tan solo conocer su identificador único (referencia directa insegura a objetos)

- Acceder a APIs con controles de acceso inexistentes para los métodos POST, PUT y DELETE.
- Elevación de privilegios. Actuar como usuario sin haber iniciado sesión o actuar como administrador cuando se inició sesión como usuario regular.
- Manipulación de metadatos, como reutilizar o modificar un token de control de acceso JSON Web Token (JWT), una cookie o un campo oculto, manipulándolos para elevar privilegios o abusar de la invalidación de tokens JWT.
- Configuraciones incorrectas de CORS (uso compartido de recursos de origen cruzado) que permiten el acceso a APIs desde orígenes no autorizados o confiables.
- Forzar la navegación a páginas autenticadas siendo usuario no autenticado o a páginas privilegiadas siendo usuario regular.

Cómo se previene

El control de acceso solo es efectivo si es implementado en el servidor (server-side) o en la API (caso serverless), donde el atacante no puede modificarlo ni manipular metadatos.

- A excepción de los recursos públicos, denegar por defecto.
- Implemente mecanismos de control de acceso una única vez y reutilícelos en toda la aplicación, incluyendo la minimización del uso de CORS.
- El control de acceso debe implementar su cumplimiento a nivel de dato y no permitir que el usuario pueda crear, leer, actulizar o borrar cualquier dato.
- Los modelos de dominio deben hacer cumplir los requisitos únicos de límite de negocio de aplicaciones.
- Deshabilite el listado de directorios del servidor web y asegúrese de que los archivos de metadatos (por ejemplo una carpeta .git) y archivos de respaldo no puedan ser accedidos a partir de la raíz del sitio web.
- Registre las fallas de control de acceso (loggin), alertando a los administradores cuando sea apropiado (por ejemplo, fallas repetidas).

- Establezca límites a la tasa de accesos permitidos a APIs y controladores de forma de poder minimizar el daño provocado por herramientas automatizadas de ataque.
- Los identificadores de sesiones deben invalidarse en el servidor luego de cerrar la sesión. Los tokens JWT deberían ser preferiblemente de corta duración para minimizar la ventana de oportunidad de ataque. Para tokens JWT de mayor duración, es sumamente recomendable seguir los estándares de OAuth de revocación de acceso.

Tanto desarrolladores como personal de control de calidad deben incluir pruebas funcionales de control de acceso tanto a nivel unitario como de integración.

A02:2021 – Fallas Criptográficas

Resumen

Subiendo una posición al número 2, anteriormente conocido como *Exposición de datos sensibles*, que es más un amplio síntoma que una causa raíz, la atención se centra en las fallas relacionadas con la criptografía (o la falta de ésta). Esto a menudo conduce a la exposición de datos sensibles. Las CWE incluidas son *CWE-259: Uso de contraseña en código fuente*, *CWE-327: Algoritmo criptográfico vulnerado o inseguro* y *CWE-331: Entropía insuficiente*.

Descripción

Lo primero es determinar las necesidades de protección de los datos en tránsito y en reposo. Por ejemplo, contraseñas, números de tarjetas de crédito, registros médicos, información personal y secretos comerciales requieren protección adicional, principalmente si están sujetos a leyes de privacidad (por ejemplo, el Reglamento General de Protección de Datos -GDPR- de la UE), o regulaciones, (por ejemplo, protección de datos financieros como el Estándar de Seguridad de Datos de PCI -PCI DSS-). Para todos esos datos:

- ¿Se transmiten datos en texto claro? Esto se refiere a protocolos como HTTP, SMTP, FTP que también utilizan actualizaciones de TLS como

STARTTLS. El tráfico externo de Internet es peligroso. Verifique todo el tráfico interno, por ejemplo, entre balanceadores de carga, servidores web o sistemas de back-end.

- ¿Se utilizan algoritmos o protocolos criptográficos antiguos o débiles de forma predeterminada o en código antiguo?
- ¿Se utilizan claves criptográficas predeterminadas, se generan o reutilizan claves criptográficas débiles, o es inexistente la gestión o rotación de claves adecuadas? ¿Se incluyen las claves criptográficas en los repositorios de código fuente?
- ¿No es forzado el cifrado, por ejemplo, faltan las directivas de seguridad de los encabezados HTTP (navegador) o los encabezados?
- ¿El certificado de servidor recibido y la cadena de confianza se encuentran debidamente validados?
- ¿Los vectores de inicialización se ignoran, se reutilizan o no se generan de forma suficientemente seguros para el modo de operación criptográfico? ¿Se utiliza un modo de funcionamiento inseguro como el ECB? ¿Se utiliza un cifrado cuando el cifrado autenticado es más apropiada?
- ¿Las contraseñas se utilizan como claves criptográficas en ausencia de una función de derivación de claves a partir de contraseñas?
- ¿Se utiliza con fines criptográficos generadores de aleatoriedad que no fueron diseñaron para dicho fin? Incluso si se elige la función correcta, debe ser inicializada (seed) por el desarrollador y, de no ser así, ¿el desarrollador ha sobrescrito la funcionalidad de semilla fuerte incorporada con una semilla que carece de suficiente entropía/imprevisibilidad?
- ¿Se utilizan funciones hash en obsoletas, como MD5 o SHA1, o se utilizan funciones hash no criptográficas cuando se necesitan funciones hash criptográficas?
- ¿Se utilizan métodos criptográficos de relleno(padding) obsoletos, como PKCS número 1 v1.5?
- ¿Se pueden explotar los mensajes de errores criptográficos como un canal lateral, por ejemplo, en forma de ataques de criptoanálisis por modificación relleno (Oracle Padding)?

Consulte ASVS Crypto (V7), Data Protection (V9) y SSL/TLS (V10)

Cómo se previene

Haga lo siguiente como mínimo, y consulte las referencias:

- Clasifique los datos procesados, almacenados o transmitidos por una aplicación. Identifique qué datos son confidenciales de acuerdo con las leyes de privacidad, los requisitos reglamentarios o las necesidades comerciales.
- No almacene datos sensibles innecesariamente. Descártelos lo antes posible o utilice una utilización de tokens compatible con PCI DSS o incluso el truncamiento. Los datos que no se conservan no se pueden robar.
- Asegúrese de cifrar todos los datos sensibles en reposo (almacenamiento).
- Garantice la implementación de algoritmos, protocolos y claves que utilicen estándares sólidos y actualizados; utilice una gestión de claves adecuada.
- Cifre todos los datos en tránsito con protocolos seguros como TLS con cifradores de confidencialidad adelantada (forward secrecy, o FS), priorización de cifradores por parte del servidor y parámetros seguros. Aplique el cifrado mediante directivas como HTTP Strict Transport Security (HSTS).
- Deshabilite el almacenamiento en caché para respuestas que contengan datos sensibles.
- Aplique los controles de seguridad requeridos según la clasificación de los datos.
- No utilice protocolos antiguos como FTP y SMTP para transportar datos sensibles.
- Almacene las contraseñas utilizando funciones robustas, flexibles, que utilicen sal en los hashes y use un factor de retraso (factor de trabajo), como Argon2, scrypt, bcrypt o PBKDF2.
- Elija vectores de inicialización apropiados para el modo de operación. Para muchos modos, esto significa usar un CSPRNG (generador de números pseudoaleatorios criptográficamente seguro). Para los modos que requieren un nonce, el vector de inicialización (IV) no necesita un CSPRNG. En todos los casos, el IV nunca debe usarse dos veces para una clave fija.

- Utilice siempre cifrado autenticado en lugar de solo cifrado.
- Las claves deben generarse criptográficamente al azar y almacenarse en la memoria como arrays de bytes. Si se utiliza una contraseña, debe convertirse en una clave mediante una función adecuada de derivación de claves basada en contraseña.
- Asegúrese de que se utilice la aleatoriedad criptográfica cuando sea apropiado y que no se utilice una semilla de una manera predecible o con baja entropía. La mayoría de las API modernas no requieren que el desarrollador genere el CSPRNG para obtener seguridad.
- Evite las funciones criptográficas y los esquemas de relleno(padding) en desuso, como MD5, SHA1, PKCS número 1 v1.5.
- Verifique de forma independiente la efectividad de la configuración y los ajustes

A03:2021 – Inyección

Resumen

La Inyección desciende a la tercera posición. El 94% de las aplicaciones fueron probadas para algún tipo de inyección con una tasa de incidencia máxima del 19%, una tasa de incidencia promedio del 3% y 274.000 ocurrencias. Las CWE incluidas son *CWE-79: Secuencia de Comandos en Sitios Cruzados (XSS)*, *CWE-89: Inyección SQL*, y la *CWE-73:Control Externo de Nombre de archivos o ruta*.

Descripción

Una aplicación es vulnerable a estos tipos de ataque cuando:

- Los datos proporcionados por el usuario no son validados, filtrados ni sanitizados por la aplicación.

- Se invocan consultas dinámicas o no parametrizadas, sin codificar los parámetros de forma acorde al contexto.
- Se utilizan datos dañinos dentro de los parámetros de búsqueda en consultas Object-Relational Mapping (ORM), para extraer registros adicionales sensibles.
- Se utilizan datos dañinos directamente o se concatenan, de modo que el SQL o comando resultante contiene datos y estructuras con consultas dinámicas, comandos o procedimientos almacenados.

Algunas de las inyecciones más comunes son SQL, NoSQL, comandos de sistema operativo, Object-Relational Mapping (ORM), LDAP, expresiones de lenguaje u Object Graph Navigation Library (OGNL). El concepto es idéntico para todos los intérpretes. La revisión del código fuente es el mejor método para detectar si las aplicaciones son vulnerables a inyecciones. Las pruebas automatizadas en todos los parámetros, encabezados, URL, cookies, JSON, SOAP y XML son fuertemente recomendados. Las organizaciones pueden incluir herramientas de análisis estático (SAST), dinámico (DAST) o interactivo (IAST) en sus pipelines de CI/CD con el fin de identificar fallas recientemente introducidas, antes de ser desplegadas en producción..

Cómo se previene

Prevenir inyecciones requiere separar los datos de los comandos y las consultas.

- La opción preferida es utilizar una API segura, que evite el uso de un intérprete por completo y proporcione una interfaz parametrizada o utilizar una herramienta de ORM.
  Nota:: Incluso cuando se parametrizan, los procedimientos almacenados pueden introducir una inyección SQL si el procedimiento PL/SQL o T-SQL concatena consultas y datos, o se ejecutan parámetros utilizando EXECUTE IMMEDIATE o exec().
- Implemente validaciones de entradas de datos en el servidor, utilizando "listas blancas". De todos modos, esto no es una defensa completa, ya que muchas aplicaciones requieren el uso de caracteres especiales, como en campos de texto o APIs para aplicaciones móviles.

- Para cualquier consulta dinámica restante, escape caracteres especiales utilizando la sintaxis de caracteres específica para el intérprete que se trate.
  Nota: La estructura de SQL como nombres de tabla, nombres de columna, etc. no se pueden escapar y, por lo tanto, los nombres de estructura suministrados por el usuario son peligrosos. Este es un problema común en el software de redacción de informes.
- Utilice LIMIT y otros controles SQL dentro de las consultas para evitar la fuga masiva de registros en caso de inyección SQL.

A04:2021 – Diseño Inseguro

Resumen

Una nueva categoría en la versión 2021. se centra en los riesgos relacionados con el diseño y las fallas arquitectónicas, exhortando a un mayor uso de: modelado de amenazas, patrones de diseño seguros y arquitecturas de

referencia. Como comunidad, debemos ir más allá de la codificación y adoptar actividades cruciales para obtener Seguridad por Diseño. Debemos "mover a la izquierda" del proceso de desarrollo las actividades de seguridad. Las CWE notables incluidas son *CWE-209: Generación de mensaje de error que contiene información confidencial*, *CWE-256: Almacenamiento desprotegido de credenciales*, *CWE-501: Violación de las fronteras de confianza* y *CWE-522: Credenciales protegidas insuficientemente*.

Descripción

El diseño inseguro es una categoría amplia que representa diferentes debilidades, expresadas como "diseño de control faltante o ineficaz". El diseño inseguro no es la fuente de las otras 10 categorías. Existe una diferencia entre un diseño inseguro y una implementación insegura. Distinguimos entre fallas de diseño y defectos de implementación por un motivo, difieren en la causa raíz y remediaciones. Incluso un diseño seguro puede tener defectos de implementación que conduzcan a vulnerabilidades que pueden explotarse. Un diseño inseguro no se puede arreglar con una implementación perfecta, ya que, por definición, los controles de seguridad necesarios nunca se crearon para defenderse de ataques específicos. Uno de los factores que contribuyen al diseño inseguro es la falta de perfiles de riesgo empresarial inherentes al software o sistema que se está desarrollando y, por lo tanto, la falta de determinación del nivel de diseño de seguridad que se requiere.

Gestión de requerimientos y recursos

Recopile y negocie los requerimientos para la aplicación con el negocio, incluidos los requisitos de protección relacionados con la confidencialidad, integridad, disponibilidad y autenticidad de todos los activos de datos y la lógica de negocio esperada. Tenga en cuenta qué tan expuesta estará su aplicación y si necesita segregación de funcionalidades (además del control de acceso). Recopile los requerimientos técnicos, incluidos los funcionales de seguridad y los no funcionales. Planifique y negocie que el presupuesto cubra el diseño, construcción, prueba y operación, incluyendo las actividades de seguridad.

Diseño seguro

El diseño seguro es una cultura y metodología que evalúa constantemente las amenazas y garantiza que el código esté diseñado y probado de manera sólida para prevenir métodos de ataque conocidos. El modelado de amenazas debe estar integrado en sesiones de refinamiento (o actividades similares); buscar cambios en los flujos de datos y el control de acceso u otros controles de seguridad. Durante la creación de las historias de usuario, determine el flujo correcto y los estados de falla. Asegúrese de que sean bien entendidos y acordados por las partes responsables e impactadas. Analice las suposiciones y las condiciones para los flujos esperados y de falla, asegúrese de que aún sean precisos y deseables. Determine cómo validar las suposiciones y hacer cumplir las condiciones necesarias para los comportamientos adecuados. Asegúrese de que los resultados estén documentados en las historias de usuario. Aprenda de los errores y ofrezca incentivos positivos para promover mejoras. El diseño seguro no es un complemento ni una herramienta que pueda agregar al software.

Ciclo de Desarrollo Seguro (S-SDLC)

El software seguro requiere un ciclo de desarrollo seguro, alguna forma de patrón de diseño seguro, metodologías de carretera pavimentada ("paved road"), bibliotecas de componentes seguros, herramientas y modelado de amenazas. Comuníquese con sus especialistas en seguridad desde el comienzo y durante todo el proyecto, así como durante su fase de mantenimiento. Considere aprovechar el Modelo de Madurez para el Aseguramiento del Software (SAMM) para ayudar a estructurar sus esfuerzos de desarrollo de software seguro.

Cómo se previene

- Establezca y use un ciclo de desarrollo seguro apoyado en Profesionales en Seguridad de Aplicaciones para ayudarlo a evaluar y diseñar la seguridad y controles relacionados con la privacidad.
- Establezca y utilice un catálogo de patrones de diseño seguros o componentes de "camino pavimentado" listos para ser utilizados.

- Utilice el modelado de amenazas para flujos críticos de autenticación, control de acceso, lógica de negocio y todo clave.
- Integre el lenguaje y los controles de seguridad en las historias de usuario.
- Integre verificaciones de viabilidad en cada capa de su aplicación (desde el frontend al backend).
- Escriba pruebas unitarias y de integración para validar que todos los flujos críticos son resistentes al modelo de amenazas. Recopile casos de uso *y* casos de mal uso para cada capa de la aplicación.
- Separe las capas del sistema y las capas de red según las necesidades de exposición y protección.
- Separe a los tenants de manera robusta por diseño en todos los niveles.
- Limitar el consumo de recursos por usuario o servicio.

A05:2021 – Configuración de Seguridad Incorrecta

Resumen

Ascendiendo una posición desde el sexto puesto en la edición anterior, el 90% de las aplicaciones se probaron para detectar algún tipo de configuración incorrecta, con una tasa de incidencia promedio del 4,5% y más de 208.000 ocurrencias de CWEs en esta categoría de riesgo. Con mayor presencia de software altamente configurable, no es sorprendente ver que esta categoría ascendiera. Las CWE notables incluidas son *CWE-16 Configuración* y *CWE-611 Restricción incorrecta entidades externas referenciadas de XML*.

Descripción

La aplicación puede ser vulnerable si:

- Le falta el *hardening* de seguridad adecuado en cualquier parte del *stack tecnológico* o permisos configurados incorrectamente en los servicios en la nube.
- Tiene funciones innecesarias habilitadas o instaladas (puertos, servicios, páginas, cuentas o privilegios innecesarios, por ejemplo).
- Las cuentas predeterminadas y sus contraseñas aún están habilitadas y sin cambios.
- El manejo de errores revela a los usuarios rastros de pila u otros mensajes de error demasiado informativos.
- Para sistemas actualizados, las últimas funciones de seguridad están deshabilitadas o no configuradas de forma segura.
- Las configuraciones de seguridad en los servidores de aplicaciones, frameworks de aplicaciones (Struts, Spring o ASP.NET por ejemplo), bibliotecas, bases de datos, etc., no poseen configurados valores seguros.
- El servidor no envía encabezados o directivas de seguridad, o no poseen configurados valores seguros.
- El software está desactualizado o es vulnerable (consulte A06:2021-Componentes Vulnerables y Desactualizados).

Sin un proceso de configuración de seguridad de aplicaciones coordinado y repetible, los sistemas corren un mayor riesgo.

Cómo se previene

Deben implementarse procesos de instalación seguros, incluyendo:

- Un proceso de *hardening* repetible agiliza y facilita la implementación de otro entorno que esté debidamente inaccesible. Los entornos de desarrollo, control de calidad y producción deben configurarse de forma idéntica, con diferentes credenciales utilizadas en cada uno. Este proceso debe automatizarse para minimizar el esfuerzo necesario para configurar un nuevo entorno seguro.
- Una plataforma mínima sin funciones, componentes, documentación ni ejemplos innecesarios. Elimine o no instale características y frameworks no utilizados.
- Una tarea para revisar y actualizar las configuraciones apropiadas para todas las notas de seguridad, actualizaciones y parches como parte del proceso de administración de parches (consulte A06: 2021-Componentes Vulnerables y Desactualizados). Revise los permisos de almacenamiento en la nube (por ejemplo, Permisos de bucket de S3).
- Una arquitectura de aplicación segmentada proporciona una separación efectiva y segura entre componentes o instancias, con segmentación, organización en contenedores o grupos de seguridad en la nube (ACLs).
- Envío de directivas de seguridad a los clientes, por ejemplo, encabezados de seguridad.
- Un proceso automatizado para verificar la efectividad de las configuraciones y ajustes en todos los entornos.

A06:2021 – Componentes Vulnerables y Desactualizados

Factores

Resumen

Era el segundo de la encuesta de la comunidad Top 10, pero también tuvo datos suficientes para llegar al Top 10 a través del análisis de datos. Los componentes vulnerables son un problema conocido que es difícil de probar y evaluar el riesgo. Es la única categoría que no tiene enumeraciones de debilidades comunes (CWE) asignadas a las CWE incluidas, por lo que se utiliza un peso de impacto/exploits predeterminado de 5,0. Las CWE notables incluidas son *CWE-1104: Uso de componentes de terceros no mantenidos* y las dos CWE del OWASP Top 10 2013 y 2017.

Descripción

Usted probablemente sea vulnerable:

- Si no conoce las versiones de todos los componentes que utiliza (tanto en el cliente como en el servidor). Esto incluye los componentes que usa directamente, así como las dependencias anidadas.
- Si el software es vulnerable, carece de soporte o no está actualizado. Esto incluye el sistema operativo, el servidor web/de aplicaciones, el sistema de administración de bases de datos (DBMS), las aplicaciones, las API y todos los componentes, los entornos de ejecución y las bibliotecas.
- Si no analiza en búsqueda de vulnerabilidades de forma regular y no se suscribe a los boletines de seguridad relacionados con los componentes que utiliza.
- Si no repara o actualiza la plataforma subyacente, frameworks y dependencias de manera oportuna y basada en el riesgo. Esto suele ocurrir en entornos en los que la aplicación de parches de seguridad es una tarea

mensual o trimestral bajo control de cambios, lo que deja a las organizaciones abiertas a días o meses de exposición innecesaria a vulnerabilidades con soluciones disponibles.

- Si los desarrolladores de software no testean la compatibilidad de las bibliotecas actualizadas, actualizadas o parcheadas.
- Si no asegura las configuraciones de los componentes (consulte A05:2021 – Configuración de Seguridad Incorrecta).

Cómo se previene

Debe existir un proceso de administración de parches que:

- Elimine las dependencias que no son utilizadas, funcionalidades, componentes, archivos y documentación innecesarios.
- Realice un inventario continuo de las versiones de los componentes en el cliente y en el servidor (por ejemplo, frameworks, bibliotecas) y sus dependencias utilizando herramientas como: versions, OWASP Dependency Check, retire.js, etc. Supervise continuamente fuentes como Common Vulnerability and Exposures (CVE) y National Vulnerability Database (NVD) para detectar vulnerabilidades en los componentes. Utilice herramientas de análisis de composición de software para automatizar el proceso. Suscríbase para recibir alertas por correo electrónico sobre vulnerabilidades de seguridad relacionadas con los componentes que utiliza.
- Solo obtenga componentes de fuentes oficiales a través de enlaces seguros. Prefiera los paquetes firmados para reducir la posibilidad de incluir un componente malicioso modificado (consulte A08:2021 – Fallas en el Software y en la Integridad de los Datos).
- Supervise las bibliotecas y los componentes que no sea mantenidos o no generen parches de seguridad para versiones anteriores. Si la aplicación de parches no es posible, considere implementar un parche virtual para monitorear, detectar o protegerse contra el problema descubierto.

Toda organización debe garantizar un plan continuo para monitorear, clasificar y aplicar actualizaciones o cambios de configuración durante la vida útil de la aplicación o portafolio de aplicaciones.

A07:2021 – Fallas de Identificación y Autenticación

Resumen

Previamente denominada como *Pérdida de Autenticación*, descendió desde la segunda posición, y ahora incluye CWEs que están más relacionados con fallas de identificación. Las CWE notables incluidas son *CWE-297: Validación incorrecta de Certificado con discrepancia de host*, *CWE-287: Autenticación incorrecta* y *CWE-384: Fijación de sesiones*.

Descripción

La confirmación de la identidad, la autenticación y la gestión de sesiones del usuario son fundamentales para protegerse contra ataques relacionados con la autenticación. Puede haber debilidades de autenticación si la aplicación:

- Permite ataques automatizados como la reutilización de credenciales conocidas, donde el atacante posee una lista de pares de usuario y contraseña válidos.
- Permite ataques de fuerza bruta u otros ataques automatizados.
- Permite contraseñas por defecto, débiles o bien conocidas, como "Password1" o "admin/admin".
- Posee procesos débiles o no efectivos para las funcionalidades de olvido de contraseña o recuperación de credenciales, como "respuestas basadas en el conocimiento", las cuales no se pueden implementar de forma segura.

- Almacena las contraseñas en texto claro, cifradas o utilizando funciones de hash débiles (consulte A02: 2021-Fallas Criptográficas).
- No posee una autenticación multi-factor o la implementada es ineficaz.
- Expone el identificador de sesión en la URL.
- Reutiliza el identificador de sesión después de iniciar sesión.
- No inválida correctamente los identificadores de sesión. Las sesiones de usuario o los tokens de autenticación (principalmente tokens de inicio de sesión único (SSO)) no son correctamente invalidados durante el cierre de sesión o luego de un período de inactividad.

Cómo se previene

- Cuando sea posible, implemente la autenticación multi-factor para evitar ataques automatizados de reutilización de credenciales conocidas, fuerza bruta y reúso de credenciales robadas.
- No incluya o implemente en su software credenciales por defecto, particularmente para usuarios administradores.
- Implemente un control contra contraseñas débiles, tal como verificar que una nueva contraseña o la utilizada en el cambio de contraseña no esté incluida en la lista de las 10,000 peores contraseñas.
- Alinear las políticas de largo, complejidad y rotación de las contraseñas con las pautas de la sección 5.1.1 para Secretos Memorizados de la guía del NIST 800-63b u otras políticas de contraseñas modernas, basadas en evidencias.
- Asegúrese que el registro, la recuperación de las credenciales y el uso de APIs, no permiten los ataques de enumeración de usuarios, mediante la utilización de los mismos mensajes genéricos en todas las salidas.
- Limite o incremente el tiempo de espera entre intentos fallidos de inicio de sesión, pero tenga cuidado de no crear un escenario de denegación de servicio. Registre todos los fallos y avise a los administradores cuando se detecten ataques de rellenos automatizados de credenciales, fuerza bruta u otros.
- Utilice un gestor de sesión en el servidor, integrado, seguro y que genere un nuevo ID de sesión aleatorio con alta entropía después de iniciar sesión.

Los identificadores de sesión no deben incluirse en la URL, deben almacenarse de forma segura y deben ser invalidados después del cierre de sesión, luego de un tiempo de inactividad o por un tiempo de espera absoluto.

A08:2021 – Fallas en el Software y en la Integridad de los Datos

Resumen

Una nueva categoría en la versión 2021 que se centra en hacer suposiciones relacionadas con las actualizaciones de software, los datos críticos y los pipelines de CI/CD sin verificación de integridad. Corresponde a uno de los mayores impactos según los sistemas de ponderación de vulnerabilidades (CVE/CVSS, siglas en inglés para Common Vulnerability and Exposures/Common Vulnerability Scoring System). Entre estos, se destacan las siguientes CWEs: *CWE-829: Inclusión de funcionalidades provenientes de fuera de la zona de confianza*, *CWE-494: Ausencia de verificación de integridad en el código descargado*, y *CWE-502: Deserialización de datos no confiables*.

Descripción

Los fallos de integridad del software y de los datos están relacionados con código e infraestructura no protegidos contra alteraciones (integridad). Ejemplos de esto son cuando una aplicación depende de plugins, bibliotecas o módulos de fuentes, repositorios o redes de entrega de contenidos (CDN) no confiables. Un pipeline CI/CD inseguro puede conducir a accesos no

autorizados, la inclusión de código malicioso o el compromiso del sistema en general. Además, es común en la actualidad que las aplicaciones implementen funcionalidades de actualización, a través de las cuales se descargan nuevas versiones de la misma sin las debidas verificaciones integridad que fueron realizadas previamente al instalar la aplicación. Los atacantes potencialmente pueden cargar sus propias actualizaciones para que sean distribuidas y ejecutadas en todas las instalaciones. Otro ejemplo es cuando objetos o datos son codificados o serializados en estructuras que un atacante puede ver y modificar, produciéndose una deserialización insegura.

Cómo se previene

- Utilice firmas digitales o mecanismos similares para verificar que el software o datos provienen efectivamente de la fuente esperada y no fueron alterados.
- Asegúrese que las bibliotecas y dependencias, tales como npm o maven son utilizadas desde repositorios confiables. Si su perfil de riesgo es alto, considere alojarlas en un repositorio interno cuyo contenido ha sido previamente analizado.
- Asegúrese que se utilice una herramienta de análisis de componentes de terceros, cómo OWASP Dependency Check u OWASP CycloneDX, con el fin de verificar la ausencia de vulnerabilidades conocidas.
- Asegúrese que se utilice un proceso de revisión de cambios de código y configuraciones para minimizar las posibilidades de que código o configuraciones maliciosas sean introducidos en su pipeline.
- Asegúrese que su pipeline CI/CD posee adecuados controles de acceso, segregación y configuraciones que permitan asegurar la integridad del código a través del proceso de build y despliegue.
- Asegúrese que datos sin cifrar o firmar no son enviados a clientes no confiables sin alguna forma de verificación de integridad o firma electrónica con el fin de detectar modificaciones o la reutilización de datos previamente serializados.

A09:2021 – Fallas en el Registro y Monitoreo

Resumen

Monitoreo y registro de seguridad provienen de la encuesta de la comunidad (tercer lugar), subió levemente desde la décima posición en el Top 10 2017. El registro y monitoreo pueden ser desafiantes para ser testeados, implicando entrevistas o preguntar si los ataques fueron detectados durante las pruebas de penetración. No hay muchos datos de CVE/CVSS para esta categoría, pero detectar y responder a las brechas es crítico. Aún así, puede tener un gran impacto para la auditabilidad, visibilidad, alertas de incidentes y análisis forense. Esta categoría se expande más allá de *CWE-117 Neutralización de salida incorrecta para registros*, *CWE-223 Omisión de información relevante para la seguridad*, y *CWE-532 Inserción de información sensible en archivo de registro*.

Descripción

Volviendo al OWASP Top 10 2021, la intención es apoyar la detección, escalamiento y respuesta ante brechas activas. Sin registros y monitoreo, las brechas no pueden ser detectadas. Registros, detecciones, monitoreo y respuesta activas insuficientes pueden ocurrir en cualquier momento:

- Eventos auditables, tales como los inicios de sesión, fallas en el inicio de sesión y transacciones de alto valor no son registradas.
- Advertencias y errores generan registros poco claros, inadecuados y en algunos casos ni se generan.
- Registros en aplicaciones y API no son monitoreados para detectar actividades sospechosas.
- Los registros son únicamente almacenados en forma local.
- Los umbrales de alerta y procesos de escalamiento no están correctamente implementados o no son efectivos.
- Las pruebas de penetración y los escaneos utilizando herramientas de pruebas dinámicas de seguridad en aplicaciones (como ser OWASP ZAP) no generan alertas.
- Las aplicaciones no logran detectar, escalar, o alertar sobre ataques activos en tiempo real ni cercanos al tiempo real.

Se es vulnerable a la fuga de información haciendo registros y eventos de alertas que sean visibles para un usuario o un atacante (consulte A01: 2021-Pérdida de Control de Acceso).

Cómo se previene

Los desarrolladores deberían implementar algunos o todos los siguientes controles, dependiendo del riesgo de la aplicación:

- Asegúrese de que todos los errores de inicio de sesión, de control de acceso y de validación de entradas de datos del lado del servidor se pueden registrar con suficiente contexto como para identificar cuentas sospechosas o maliciosas y mantenerlo durante el tiempo suficiente para permitir un posterior análisis forense.
- Asegúrese de que los registros se generen en un formato fácil de procesar por las herramientas de gestión de registros.

- Asegúrese de que los datos de registros son correctamente codificados para prevenir inyecciones o ataques en el sistema de monitoreo o registros.
- Asegúrese de que las transacciones de alto valor poseen una traza de auditoria con controles de integridad para evitar la modificación o el borrado, tales como permitir únicamente la inserción en las tablas de base de datos o similares.
- Los equipos de DevSecOps debe establecer alertas y monitoreo efectivo tal que se detecte actividades sospechosas y responderlas rápidamente.
- Establezca o adopte un plan de respuesta y recuperación, tal como NIST 800-61r2 o posterior.

Existen frameworks de protección de aplicaciones comerciales y de código abierto, tales como el conjunto de reglas de ModSecurity de OWASP y el conjunto de programas de correlación de registros de código abierto como ser ELK (Elasticsearch, Logstash, Kibana) con paneles personalizados y alertas.

A10:2021 – Falsificación de Solicitudes del Lado del Servidor (SSRF)

Resumen

Esta categoría se agrega debido a la encuesta de la comunidad Top 10 (primer lugar). Los datos muestran una tasa de incidencia relativamente baja con una cobertura de pruebas por encima del promedio, junto con calificaciones por encima del promedio para la capacidad de explotación e impacto. Como es probable que estas nuevas entradas sean una única o un pequeño grupo de Enumeraciones de debilidades comunes (CWE) para tomar en cuenta y concientizar sobre ellas, la esperanza es que se enfoque la atención en ellas y puedan integrarse en una categoría más amplia en una edición futura.

Descripción

Las fallas de SSRF ocurren cuando una aplicación web está obteniendo un recurso remoto sin validar la URL proporcionada por el usuario. Permite que un atacante coaccione a la aplicación para que envíe una solicitud falsificada a un destino inesperado, incluso cuando está protegido por un firewall, VPN u otro tipo de lista de control de acceso a la red (ACL).

Dado que las aplicaciones web modernas brindan a los usuarios finales funciones convenientes, la búsqueda de una URL se convierte en un escenario común. Como resultado, la incidencia de SSRF está aumentando. Además, la gravedad de SSRF es cada vez mayor debido a los servicios en la nube y la complejidad de las arquitecturas.

Cómo se previene

Los desarrolladores pueden prevenir SSRF implementando algunos o todos los siguientes controles de defensa en profundidad:

Desde la capa de red

- Segmente la funcionalidad de acceso a recursos remotos en redes separadas para reducir el impacto de SSRF
- Haga cumplir las políticas de firewall "denegar por defecto" o las reglas de control de acceso a la red para bloquear todo el tráfico de la intranet excepto el esencial.
  Consejos:
  ~ Establezca la propiedad y un ciclo de vida para las reglas de firewall basadas en aplicaciones.
  ~ Registre en logs todos los flujos de red aceptados y bloqueados en firewalls (consulte A09: 2021-Fallas en el Registro y Monitoreo).

Desde la capa de aplicación:

- Sanitice y valide todos los datos de entrada proporcionados por el cliente
- Haga cumplir el esquema de URL, el puerto y destino a través de una lista positiva de items permitidos
- No envíe respuestas en formato "crudo" a los clientes
- Deshabilite las redirecciones HTTP

- Tenga en cuenta la coherencia de la URL para evitar ataques como el enlace de DNS y las condiciones de carrera de "tiempo de verificación, tiempo de uso" (TOCTOU por sus siglas en inglés)

No mitigue SSRF mediante el uso de una lista de denegación o una expresión regular. Los atacantes poseen listas de payloads, herramientas y habilidades para eludir las listas de denegación.

Medidas adicionales a considerar:

- No implemente otros servicios relevantes para la seguridad en los sistemas frontales (por ejemplo, OpenID). Controle el tráfico local en estos sistemas (por ejemplo, localhost)
- Para frontends con grupos de usuarios dedicados y manejables, use el cifrado de red (por ejemplo, VPN) en sistemas independientes para