Introducción a la Programación y Computación 1 Sección E

Ing. MSc. Neftalí Calderón

Pérdida de control de acceso

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.

Cómo prevenir:

- A excepción de los recursos públicos, denegar por defecto.
- Implemente mecanismos de control de acceso una única vez y reutilícelos en toda la aplicación
- El control de acceso debe implementar su cumplimiento a nivel de dato y no permitir que el usuario pueda crear, leer, actualizar o borrar cualquier dato.
- 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).
- Los identificadores de sesiones deben invalidarse en el servidor luego de cerrar la sesión.

Fallas criptográficas

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.

Cómo prevenir:

- 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.
- Asegúrese de cifrar todos los datos sensibles que estan almacenados.
- 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.
- Deshabilite el almacenamiento en caché para respuestas que contengan datos sensibles.
- No utilice protocolos antiguos como FTP y SMTP para transportar datos sensibles.
- Almacene las contraseñas utilizando funciones robustas, flexibles, que utilicen saltos en los hashes y use un factor de retraso.
- Utilice siempre cifrado autenticado en lugar de solo cifrado.

Inyección de código

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.

Cómo prevenir:

- La opción preferida es utilizar una API segura, que evite el uso de un intérprete por completo y proporcione una interfaz parametrizada.
- 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.
- Para cualquier consulta dinámica restante, escape caracteres especiales utilizando la sintaxis de caracteres específica para el intérprete que se trate.
- 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.

Diseño inseguro

El diseño inseguro es una categoría amplia que representa diferentes debilidades, expresadas como "diseño de control faltante o ineficaz". 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.

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.
- 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.

Configuración de seguridad incorrecta

Tiene funciones innecesarias habilitadas o instaladas (puertos, servicios, páginas, cuentas o privilegios innecesarios)
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.

Cómo se previene:
- Una plataforma mínima sin funciones, componentes, documentación ni ejemplos innecesarios. Elimine o no instale características y frameworks no utilizados.
- Revise los permisos de almacenamiento en la nube.
- 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.

Componentes vulnerables o desactualizados

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.

Cómo prevenir:

- 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.
- Solo obtenga componentes de fuentes oficiales a través de enlaces seguros.
- Supervise las bibliotecas y los componentes que no sea mantenidos o no generen parches de seguridad para versiones anteriores.

Fallas de identificación y autenticación

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.

Como prevenir:

- 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.
- Limite o incremente el tiempo de espera entre intentos fallidos de inicio de sesión.
- 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.

Fallas en el software y la integridad de los datos

Los fallos de integridad del software y de los datos están relacionados con código e infraestructura no protegidos contra alteraciones (integridad).
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 prevenir:
- 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 se utilice una herramienta de análisis de componentes de terceros.
- 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.
- 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.

Falsificación de solicitudes del lado del servidor

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).