2023

Hugo Gómez Sánchez

Inetum

28-3-2023

Vulnerabilidades Web



Contenido

[Vulnerabilidad → Inyección SQL 3](#_Toc130988379)

[Descripción 4](#_Toc130988380)

[Mitigación 4](#_Toc130988381)

[Referencias 4](#_Toc130988382)

[Vulnerabilidad → Cross-Site Scripting 3](#_Toc130988383)

[Descripción 4](#_Toc130988384)

[Mitigación 4](#_Toc130988385)

[Referencias 4](#_Toc130988386)

[Vulnerabilidad → Cross-Site Request Forgery 3](#_Toc130988387)

[Descripción 4](#_Toc130988388)

[Mitigación 4](#_Toc130988389)

[Referencias 4](#_Toc130988390)

[Vulnerabilidad → Detección de directorios 3](#_Toc130988391)

[Descripción 4](#_Toc130988392)

[Mitigación 4](#_Toc130988393)

[Referencias 4](#_Toc130988394)

[Vulnerabilidad → desbordamiento de búfer en PHP 3](#_Toc130988395)

[Descripción 4](#_Toc130988396)

[Mitigación 4](#_Toc130988397)

[Referencias 4](#_Toc130988398)

[Vulnerabilidad → Blind XPath Injection 3](#_Toc130988399)

[Descripción 4](#_Toc130988400)

[Mitigación 4](#_Toc130988401)

[Referencias 4](#_Toc130988402)

[Vulnerabilidad → Content Spoofing 3](#_Toc130988403)

[Descripción 4](#_Toc130988404)

[Mitigación 4](#_Toc130988405)

[Referencias 4](#_Toc130988406)

[Vulnerabilidad → Clickjacking 3](#_Toc130988407)

[Descripción 4](#_Toc130988408)

[Mitigación 4](#_Toc130988409)

[Referencias 4](#_Toc130988410)

[Vulnerabilidad → JSON injection 3](#_Toc130988411)

[Descripción 4](#_Toc130988412)

[Mitigación 4](#_Toc130988413)

[Referencias 4](#_Toc130988414)

[Vulnerabilidad → SPYWARE 3](#_Toc130988415)

[Descripción 4](#_Toc130988416)

[Vulnerabilidad → Broken cryptography 3](#_Toc130988417)

[Descripción 4](#_Toc130988418)

[Mitigación 4](#_Toc130988419)

[Referencias 4](#_Toc130988420)

[Vulnerabilidad → Insecure random number 3](#_Toc130988421)

[Descripción 4](#_Toc130988422)

[Vulnerabilidad → Forced browsing 3](#_Toc130988423)

[Descripción 4](#_Toc130988424)

[Mitigación 4](#_Toc130988425)

[Referencias 4](#_Toc130988426)

[Vulnerabilidad → Direct Dynamic Code Evaluation - Eval Injection 3](#_Toc130988427)

[Descripción 3](#_Toc130988428)

[Mitigación 3](#_Toc130988429)

[Referencias 3](#_Toc130988430)

# **Vulnerabilidad → Inyección SQL**

## Descripción

Esta vulnerabilidad utiliza el lenguaje de base de datos SQL, consiste en lanzar una sentencia SQL maliciosa a la base de datos a través de algún elemento de la web en el que se pueda introducir texto el cliente web procesara la consulta dejando así vulnerable la base de datos.

Esto significa que una persona sin permisos puede obtener información confidencial, modificar o eliminar datos.

## Mitigación

1. El uso de expresiones regulares para asegurarse de que la entrada cumple con el formato adecuado.
2. Probar tu sistema de seguridad para buscar entradas que permitan hacer inyección SQL es una fase importante de la defensa de tu sistema.
3. Se deben banear los caracteres especiales que pueden ser usados en una inyección SQL, esto se puede lograr con los métodos de JavaScript encodeURI() o encodeURIComponent()
4. Utilizar parámetros de consulta en la que los valores son reemplazados por variables hace que la consulta sea más segura.

## Referencias

-[Avast Academy](https://www.avast.com/es-es/c-sql-injection#:~:text=La%20inyección%20de%20SQL%20es,la%20información%20de%20los%20usuarios.)

-[Software Lab](https://softwarelab.org/es/que-es-inyeccion-sql/)

# **Vulnerabilidad → Cross-Site Scripting**

## Descripción

La manera en la que el atacante explota esta vulnerabilidad es creando un script e intentando que el cliente web lo ejecute. Este tipo de ataques es similar a la inyección SQL ya que se busca una entrada de texto para poder escribir el script dentro. Estos scripts están escritos en JavaScript para que el buscador los pueda interpretar. Dependiendo de cómo esté configurado el cliente web los scripts podrían quedarse almacenados en una base de datos común para todos los usuarios de la web, provocando así que cada vez que alguien cargue la web se ejecute dicho script en su navegador.

¡Imagina que el atacante crea un script para robarte las cookies de sesión!

## Mitigación

1. Es esencial utilizar expresiones regulares para validar que el usuario no esté ingresando sintaxis de JavaScript en este tipo de ataques, al igual que en la inyección SQL.
2. Incluir encabezados HTTP, como X-XSS-Protection, es crucial para prevenir el XSS, ya que estos encabezados informan a los navegadores cómo manejar la ejecución de scripts en una página web.
3. Se deben banear los caracteres especiales que pueden ser usados en una inyección SQL, esto se puede lograr con los métodos de JavaScript encodeURI() o encodeURIComponent()

## Referencias

-[Youtube Video](https://www.youtube.com/watch?v=EWGUznyQIhE)

-[OWASP](https://owasp.org/www-community/attacks/xss/#:~:text=Cross-Site%20Scripting%20(XSS),to%20a%20different%20end%20user.)

# **Vulnerabilidad → Cross-Site Request Forgery**

## Descripción

Esta vulnerabilidad consiste en que el atacante crea una página web maliciosa que ejecuta un método post del protocolo http a la web en la que tienes iniciado sesión, provocando así alteraciones que van desde borrar tu cuenta de dicha página hasta robar tus datos o recursos de esta.

## Mitigación

1. Los recursos o uris que tienes en tú página web no deben de ser verbos
2. Utilizar tokens CSRF: Los tokens CSRF son códigos aleatorios generados por el servidor y enviados al navegador del usuario. El navegador debe incluir este token en todas las solicitudes HTTP posteriores para validar que la solicitud proviene del sitio web correcto. Si el token no coincide, la solicitud será rechazada.
3. El servidor debe verificar que la solicitud HTTP proviene de un origen válido antes de procesarla.
4. Las cookies seguras solo se envían a través de conexiones HTTPS seguras.

## Referencias

-[Youtube Video](https://www.youtube.com/watch?v=eWEgUcHPle0&t)

-[WIKI](https://es.wikipedia.org/wiki/Cross-site_request_forgery)

# **Vulnerabilidad → Detección de directorios**

## Descripción

Es una vulnerabilidad que permite que un atacante encuentre archivos y directorios que no deberían ser accesibles públicamente.

Esta vulnerabilidad puede ser explotada mediante el uso de técnicas de enumeración de directorios, como la fuerza bruta o la enumeración de archivos, que permiten al atacante descubrir la estructura del sistema de archivos del servidor web.

## Mitigación

Una forma de mitigar esta vulnerabilidad es desactivar la opción de listado de directorios en el servidor web. Esto se puede hacer mediante la configuración de archivos .htaccess o mediante la configuración del servidor web. También es importante asegurarse de que todos los archivos y directorios en el sitio web tengan permisos adecuados y estén protegidos contra accesos no autorizados.

## Referencias

-[OWASP](https://wiki.owasp.org/index.php/Testing_Directory_traversal/file_include_(OTG-AUTHZ-001))

# **Vulnerabilidad → desbordamiento de búfer en PHP**

## Descripción

Un atacante puede aprovechar una vulnerabilidad en el lenguaje de programación PHP para sobrecargar el búfer de memoria y ejecutar código malicioso en el servidor.

## Mitigación

Una forma de mitigar esta vulnerabilidad es aplicar parches de seguridad y actualizaciones regulares del sistema operativo y las aplicaciones web. También se recomienda evitar el uso de funciones inseguras y limitar el tamaño de entrada permitido en el servidor.

## Referencias

-[Wiki](https://es.wikipedia.org/wiki/Desbordamiento_de_búfer)

# **Vulnerabilidad → Blind XPath Injection**

## Descripción

XPath es un tipo de lenguaje de consulta que describe cómo ubicar elementos específicos.

Dado que es un lenguaje de consulta, XPath es algo similar al lenguaje de consulta estructurado (SQL), sin embargo, XPath es diferente en el sentido de que se puede usar para hacer referencia a casi cualquier parte de un documento XML sin restricciones de control de acceso.

Los ataques Blind XPath Injection se pueden utilizar para extraer datos de una aplicación que incrusta datos proporcionados por el usuario de forma no segura.

## Mitigación

1. Validación y filtrado de entradas: La validación y filtrado de todas las entradas de usuario es una medida importante para prevenir la inyección ciega de XPath. Es necesario asegurarse de que todas las entradas de usuario sean validadas y filtradas adecuadamente antes de ser utilizadas en una consulta XPath.
2. Implementación de políticas de seguridad: Las políticas de seguridad pueden ayudar a prevenir la inyección ciega de XPath. Una política de seguridad podría requerir que todas las consultas XPath sean validadas y filtradas antes de ser enviadas al servidor.
3. Deshabilitar características innecesarias: Si no se necesitan características específicas en el servidor web, se deben deshabilitar para reducir la superficie de ataque. Por ejemplo, si no se utiliza XPath en una aplicación web, se debe deshabilitar la función.

## Referencias

-[OWASP](https://owasp.org/www-community/attacks/Blind_XPath_Injection)

# **Vulnerabilidad → Content Spoofing**

## Descripción

Consiste en un ataque de ingeniería social en el que el atacante crea una versión idéntica o modificada de una web. El usuario al no percatarse de esto puede poner en peligro la integridad de sus datos personales.

Esto se puede lograr a través de técnicas como la suplantación de identidad, la manipulación de contenido, la inyección de scripts maliciosos y la falsificación de información.

## Mitigación

Validación y sanitización de entradas: Es importante validar y sanitizar todas las entradas de usuario para asegurarse de que sean seguras antes de ser utilizadas en la página web. Esto incluye validar la entrada de los formularios, las cookies y los parámetros de la URL.

Implementación de políticas de seguridad: Las políticas de seguridad pueden ayudar a prevenir el content spoofing. Por ejemplo, se pueden implementar políticas de seguridad de contenido (CSP) para restringir el tipo de contenido que se puede cargar en una página web.

Protección contra la suplantación de identidad: Se deben implementar medidas de seguridad para proteger contra la suplantación de identidad, como la autenticación de usuarios y la implementación de HTTPS para cifrar la comunicación entre el cliente y el servidor.

## Referencias

-[OWASP](https://owasp.org/www-community/attacks/Content_Spoofing)

# **Vulnerabilidad → Clickjacking**

## Descripción

Clickjacking es un ataque basado en la interfaz que engaña a los usuarios de un sitio web para que sin saberlo hagan clic en enlaces maliciosos. En clickjacking, los atacantes incrustan sus enlaces maliciosos en botones o páginas legítimas en un sitio web.

## Mitigación

Implementar X-Frame-Options: La cabecera de respuesta HTTP X-Frame-Options permite a los servidores web especificar si las páginas pueden ser mostradas en un marco o iframe. Esta medida ayuda a prevenir el clickjacking al impedir que una página se muestre en un marco o iframe no autorizado.

Usar Content Security Policy (CSP): La política de seguridad de contenido (CSP) es una cabecera HTTP que permite a los sitios web especificar los orígenes de los recursos que se pueden cargar en una página. Al limitar los orígenes permitidos, se puede evitar que se carguen recursos de sitios maliciosos o desconocidos en una página.

## Referencias

-[OWASP](https://owasp.org/www-community/attacks/Clickjacking)

# **Vulnerabilidad → JSON injection**

## Descripción

El objetivo de un ataque de JSON injection es engañar a la aplicación web para que interprete el objeto JSON malicioso como si fuera una respuesta válida del servidor. Una vez que el atacante ha logrado esto, puede ejecutar código arbitrario en el lado del cliente, como scripts maliciosos o peticiones HTTP.

## Mitigación

Para prevenir ataques de JSON injection, es importante validar y sanear los datos JSON que se reciben del servidor antes de procesarlos. Esto puede hacerse mediante la implementación de medidas de seguridad adecuadas en la aplicación, como el uso de bibliotecas de análisis de JSON seguras, la validación de entradas de usuario y la limpieza de datos maliciosos.

## Referencias

-[COMPARITECH](https://www.comparitech.com/net-admin/json-injection-guide/#:~:text=JSON%20injection%20occurs%20when%3A,using%20the%20JavaScript%20eval%20function.)

# **Vulnerabilidad → SPYWARE**

## Descripción

El spyware es un programa que captura información estadística de la computadora de un usuario y la envía a través de Internet sin la aceptación del usuario. Esta información generalmente se obtiene de las cookies y del historial del navegador web. El software espía también puede instalar otro software, mostrar anuncios o redirigir la actividad del navegador web. El spyware se diferencia de un virus, un gusano y un adware en varios aspectos. El spyware no se autorreplica ni se distribuye como los virus y los gusanos, y no necesariamente muestra anuncios como el adware.

# **Vulnerabilidad → Broken cryptography**

## Descripción

Es una vulnerabilidad de seguridad que ocurre cuando un cifrado utilizado para proteger datos se debilita o se rompe debido a un diseño inadecuado, algoritmos obsoletos, vulnerabilidades en la implementación o claves débiles.

Los ataques de cifrado roto pueden permitir a los atacantes acceder a datos confidenciales, como información de identificación personal, credenciales de inicio de sesión, contraseñas, información financiera o secretos comerciales. Además, los ataques de cifrado roto pueden permitir a los atacantes falsificar información, manipular transacciones o comprometer la integridad de los datos.

## Mitigación

Para mitigar los riesgos de "Broken cryptography", se recomienda utilizar algoritmos de cifrado fuertes y actualizados, implementar políticas de contraseñas seguras, proteger adecuadamente las claves de cifrado, y realizar pruebas de penetración para detectar y corregir vulnerabilidades de cifrado. Además, es importante estar al tanto de las últimas tendencias y amenazas de seguridad en materia de cifrado para estar preparados para proteger adecuadamente los datos sensibles.

## Referencias

-[OWASP](https://owasp.org/www-project-mobile-top-10/2014-risks/m6-broken-cryptography)

# **Vulnerabilidad → Insecure random number**

## Descripción

La generación de números aleatorios inseguros se refiere a la práctica de generar números aleatorios que no son verdaderamente aleatorios o impredecibles, lo que puede representar un riesgo de seguridad en varios contextos.

Un ejemplo común de generación de números aleatorios inseguros es el uso de un algoritmo o fórmula simple para generar números aleatorios. Por ejemplo, usar la hora del reloj o la memoria del sistema de una computadora como semilla para un generador de números pseudoaleatorios puede parecer aleatorio, pero no es realmente aleatorio y puede ser predecido por un atacante con suficiente poder de cómputo y conocimiento del algoritmo.

Otro ejemplo es el uso de una fuente de entropía débil, como el movimiento del mouse o los patrones de entrada del teclado, para generar números aleatorios. Estos patrones pueden ser predecibles y, por lo tanto, comprometer la seguridad de los sistemas que dependen de la aleatoriedad de los números generados.

# **Vulnerabilidad → Forced browsing**

## Descripción

La navegación forzada es un ataque cuyo objetivo es enumerar y acceder a recursos a los que la aplicación no hace referencia, pero que aún son accesibles.

Un atacante puede utilizar técnicas de fuerza bruta para buscar contenidos no vinculados en el directorio del dominio, como directorios y archivos temporales, y archivos de configuración y copia de seguridad antiguos. Estos recursos pueden almacenar información confidencial sobre aplicaciones web y sistemas operativos, como el código fuente, las credenciales, el direccionamiento de la red interna, etc., por lo que se consideran un recurso valioso para los intrusos.

Este ataque se realiza manualmente cuando los directorios y las páginas del índice de la aplicación se basan en la generación de números o valores predecibles, o se utilizan herramientas automatizadas para archivos comunes y nombres de directorios.

Este ataque también se conoce como ubicación de recursos predecible, enumeración de archivos, enumeración de directorios y enumeración de recursos.

## Mitigación

1. Implementar la autenticación en todas las páginas web que contengan información sensible o restringida. Esto garantizará que solo los usuarios autorizados tengan acceso a los recursos protegidos.
2. Restringir el acceso a los recursos sensibles mediante el uso de permisos y roles de usuario. Esto puede ayudar a limitar la cantidad de usuarios que pueden acceder a recursos sensibles y reducir el riesgo de exposición.
3. Implementar controles de acceso basados en sesiones para detectar y bloquear intentos de Forced Browsing. Esto puede incluir el monitoreo del tráfico de red y la identificación de patrones de acceso inusuales o sospechosos.
4. Utilizar técnicas de ofuscación de URL, como la implementación de hashes aleatorios en las URL, para dificultar la adivinación manual de las mismas.
5. Limitar la exposición de información sensible mediante la implementación de prácticas de seguridad sólidas en el diseño y desarrollo de aplicaciones web, incluyendo la validación adecuada de entradas y la gestión adecuada de errores. Esto ayudará a reducir el riesgo de exposición de información sensible a través de vulnerabilidades de seguridad.

## Referencias

-[OWASP](https://owasp.org/www-community/attacks/Forced_browsing)

# **Vulnerabilidad → Direct Dynamic Code Evaluation - Eval Injection**

## Descripción

Este ataque consiste en un script que no valida correctamente las entradas del usuario en el parámetro de la página. Un usuario remoto puede proporcionar una URL especialmente diseñada para pasar código arbitrario a una instrucción eval(), lo que da como resultado la ejecución del código.

## Mitigación

1. Validación de entrada: Todas las entradas de usuario deben ser validadas y filtradas adecuadamente para evitar la inyección de código malicioso en la aplicación. Se debe prestar especial atención a las entradas que se utilizan en la evaluación de código dinámico.
2. Uso de bibliotecas de seguridad: Es importante utilizar bibliotecas de seguridad y marcos de trabajo que puedan ayudar a mitigar el riesgo de la inyección de código Eval. Estas bibliotecas y marcos de trabajo a menudo incluyen controles de seguridad integrados para evitar la inyección de código malicioso.

## Referencias

-[OWASP](https://owasp.org/www-community/attacks/Direct_Dynamic_Code_Evaluation_Eval%20Injection)