TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TLAXIACO

SEGURIDAD Y VIRTUALIZACION

INVESTIGACIÓN

CARRERA:

INGENIERIA EN SISTEMAS COMPUTACIONALES

PRESENTA:

ANA MICHEL LEÓN LEÓN - 21620112

RUFINO MENDOZA VAZQUEZ - 21620198

ROSA SALAZAR DOROTEO - 18620216

FERNANDA RUIZ HERAS - 21520151

DOCENTE

ING. EDWARD OSORIO SALINA

Tlaxiaco, Oax., septiembre de 2024.

29/09/2024

"Educación, ciencia y tecnología, progreso día con día"®

1. ¿Qué es la inyección de SQL?

Una inyección de SQL, a veces abreviada como SQLi, es un tipo de vulnerabilidad en la que un atacante usa un trozo de código SQL (lenguaje de consulta estructurado) para manipular una base de datos y acceder a información potencialmente valiosa. Es uno de los tipos de ataques más frecuentes y amenazadores, ya que puede atacar prácticamente cualquier sitio o aplicación web que use una base de datos basada en SQL

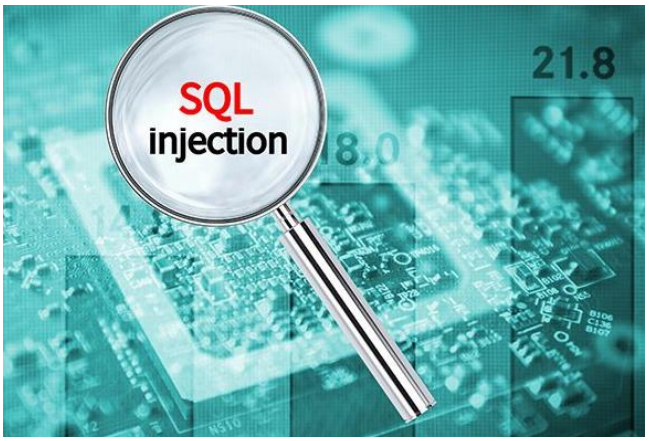

Ilustración 1_SQL inyeccion

Ejemplo de inyección de SQL:

SELECT * FROM users WHERE username = 'admin' AND password = 'password';

2. Blind SQL Injection (Inyección SQL Ciega)

La inyección SQL ciega es una variante de la inyección SQL que se utiliza cuando la aplicación no devuelve directamente los resultados de la consulta, es decir, el atacante no ve los resultados de su inyección en la página web. Sin embargo, pueden obtener información observando los cambios en el comportamiento de la aplicación (como retrasos o respuestas booleanas).

Ilustración 2_Esquema de ataque de inyeccion SQL

Existen dos tipos de inyección SQL ciega:

- ✓ Basada en booleanos: el atacante intenta adivinar la estructura de la base de datos enviando consultas que devuelven respuestas verdaderas o falsas.
- ✓ Basada en tiempo: el atacante introduce consultas que provocan un retraso en la respuesta del servidor para inferir si la consulta es verdadera o falsa.

3. SQL Injection basada en errores

La inyección SQL basada en errores es un tipo de inyección que explota los mensajes de error proporcionados por la base de datos. Estos errores, cuando no están debidamente ocultos o manejados, pueden revelar información valiosa sobre la estructura de la base de datos, como los nombres de las tablas o las columnas. Al aprovechar esta información, el atacante puede crear consultas más específicas y efectivas.

Ilustración 3_Tipo de errores

Por ejemplo, un atacante podría provocar intencionadamente un error mediante la manipulación de una consulta SQL para que la aplicación devuelva detalles técnicos sobre la base de datos.

4. SQL Injection basada en tiempo

La inyección SQL basada en tiempo es una técnica de Blind SQL Injection donde el atacante introduce comandos que provocan un retraso en la ejecución de la

consulta SQL, de modo que pueda inferir información basada en el tiempo de respuesta del servidor. Esta técnica es útil cuando la aplicación no devuelve mensajes de error ni resultados visibles para el atacante.

Características

- Interacción basada en tiempo: Utiliza funciones de la base de datos (como SLEEP en MySQL o WAITFOR DELAY en SQL Server) para provocar un retraso en la respuesta.
- Aplicable en entornos con respuestas limitadas: Ideal para situaciones donde no se reciben mensajes de error o retroalimentación visual sobre el éxito de la inyección.
- Lenta pero efectiva: Debido a que se basa en la espera, los ataques pueden ser más lentos, ya que el atacante tiene que esperar por cada respuesta.

Ventajas

Eficaz en sistemas blindados:

Esta técnica es útil cuando la aplicación no devuelve mensajes de error ni datos explícitos, por lo que puede funcionar en entornos más seguros donde los otros tipos de inyección son menos efectivos.

Difícil de detectar mediante auditorías visuales:

No provoca mensajes de error visibles ni da acceso directo a datos sensibles. Los administradores pueden no notar el ataque si no supervisan el tiempo de respuesta del servidor.

Desventajas

Ataque más lento:

Al depender de los retrasos en las respuestas del servidor, el proceso de ataque puede ser considerablemente más lento que otros métodos de inyección, especialmente si el atacante necesita hacer muchas consultas para extraer información relevante.

Más fácil de detectar con monitoreo de rendimiento:

Aunque es difícil de identificar visualmente, puede ser más detectable con herramientas de monitoreo de rendimiento de la red o el servidor, ya que introduce retrasos anómalos en las respuestas de la aplicación.

Efecto limitado si se aplican contramedidas de tiempo:

Algunas bases de datos o aplicaciones pueden tener límites en la cantidad de tiempo de espera que se puede aplicar en una consulta, lo que reduce la efectividad del ataque si los retrasos son mínimos.

5. SQL Injection en procedimientos almacenados

Los procedimientos almacenados son conjuntos de instrucciones SQL predefinidas que se almacenan en la base de datos y pueden ser ejecutadas por una aplicación. Si los procedimientos almacenados no están adecuadamente protegidos, también pueden ser vulnerables a inyecciones SQL.

Características.

Uso de parámetros dinámicos: Los procedimientos almacenados permiten ejecutar consultas SQL dinámicas utilizando parámetros de entrada, lo que los hace vulnerables si los datos proporcionados no se validan correctamente.

Mayor poder de ejecución: Los procedimientos almacenados pueden realizar operaciones más complejas y avanzadas, lo que los hace más peligrosos si se explotan.

Residen en el servidor de la base de datos: Al estar en el servidor de la base de datos, tienen un nivel de acceso elevado, lo que permite a los atacantes potencialmente realizar acciones más críticas (modificar, eliminar datos, etc.).

Ventajas

Mayor acceso a funciones críticas:

Los procedimientos almacenados a menudo tienen permisos elevados dentro de la base de datos, lo que le permite al atacante realizar acciones más dañinas si logra explotarlos, como alterar o eliminar datos importantes.

Acceso directo a la base de datos:

Como los procedimientos almacenados se ejecutan directamente en el servidor de la base de datos, no dependen de capas intermedias (como aplicaciones web), lo que reduce el riesgo de que las inyecciones sean bloqueadas por esas capas.

Desventajas.

Dificultad para explotar si el código está cerrado:

En muchos casos, los procedimientos almacenados no están expuestos directamente a los usuarios o desarrolladores externos, lo que dificulta su explotación si no se tiene visibilidad del código.

Menos común en aplicaciones modernas:

En comparación con las consultas dinámicas dentro de las aplicaciones, las inyecciones en procedimientos almacenados son menos comunes en aplicaciones modernas que utilizan frameworks ORM, que gestionan automáticamente las consultas.

Un ejemplo es un procedimiento almacenado que acepta parámetros de entrada directamente desde un usuario sin validación, lo que permite a un atacante manipular las entradas para ejecutar consultas SQL no autorizadas.

6. SQL Injection en ORM

Los ORM (Object-Relational Mappers) son herramientas que permiten a los desarrolladores interactuar con bases de datos utilizando modelos orientados a objetos. Si bien los ORM, como Hibernate o Entity Framework, suelen proteger contra inyecciones SQL, no son inmunes. Los ataques de inyección pueden ocurrir si los desarrolladores utilizan consultas SQL nativas o permiten la ejecución de comandos SQL en texto plano.

Características

Interacción basada en objetos: En lugar de escribir consultas SQL directamente, los desarrolladores interactúan con la base de datos a través de objetos que representan tablas y registros.

Consultas automáticas generadas por el ORM: El ORM convierte operaciones sobre objetos en consultas SQL automáticamente, aunque los desarrolladores aún pueden escribir SQL "nativo" en algunos casos.

Ventajas

Facilita la explotación si se usan consultas nativas:

Aunque los ORM generan consultas SQL seguras en la mayoría de los casos, el uso de consultas nativas o dinámicas en el ORM abre la posibilidad de explotar inyecciones SQL, especialmente cuando los desarrolladores pasan parámetros sin validación.

Menor percepción de vulnerabilidad:

Como los ORM tienden a ser considerados más seguros por defecto, los desarrolladores pueden no prestar tanta atención a la sanitización de entradas al escribir consultas SQL personalizadas.

Desventajas

Mayor seguridad en la generación automática de consultas:

Al usar métodos ORM estándar (como consultas basadas en objetos), las vulnerabilidades de inyección SQL son prácticamente inexistentes, ya que los ORM gestionan de forma segura la construcción de consultas SQL.

Limitado a malas prácticas de programación:

La inyección SQL en ORM es menos común que en sistemas basados únicamente en consultas SQL tradicionales, ya que requiere que los desarrolladores omitan las mejores prácticas de seguridad del ORM.

7. Herramientas para detectar y prevenir SQL Injection

Herramientas para detectar SQL Injection:

- SQLMap: Herramienta de código abierto que permite automatizar la detección y explotación de vulnerabilidades de inyección SQL.
- Havij: Una herramienta de automatización que facilita la explotación de inyecciones SQL.
- Burp Suite: Proporciona una plataforma completa para realizar pruebas de seguridad en aplicaciones web, incluida la detección de inyecciones SQL.
- OWASP ZAP (Zed Attack Proxy): Herramienta gratuita para la seguridad en aplicaciones web, que permite realizar análisis automáticos y manuales, incluida la identificación de inyecciones SQL.

Herramientas y técnicas para prevenir SQL Injection:

- ORMs seguros: Usar ORMs que manejen de forma segura las consultas SQL y protejan contra inyecciones SQL.
- Consultas preparadas (Prepared Statements): Las consultas preparadas aseguran que las entradas del usuario sean tratadas como datos y no como parte de la consulta SQL.
- Validación y escape de entradas: Verificar y filtrar las entradas proporcionadas por el usuario antes de incluirlas en las consultas SQL.
- ORMs y frameworks seguros: Como Sequelize (para Node.js) o Entity Framework (para .NET), que incluyen mecanismos para evitar la inyección SQL.
- WAF (Web Application Firewall): Firewalls diseñados para filtrar, monitorizar y bloquear solicitudes maliciosas dirigidas a aplicaciones web, incluidas las inyecciones SQL.

CONCLUSION:

En conclusión, la inyección SQL es una de las vulnerabilidades más peligrosas y comunes en aplicaciones que interactúan con bases de datos. A lo largo de los años, ha evolucionado en diferentes formas y niveles de complejidad, lo que requiere un entendimiento profundo para detectarla y prevenirla eficazmente. A pesar de las diferencias en estas técnicas, comparten ciertos principios clave: todas explotan una mala validación de las entradas y una construcción insegura de las consultas SQL. Para mitigar estos riesgos, es crucial implementar consultas parametrizadas, validación estricta de entradas, uso adecuado de ORM, y herramientas especializadas para detectar y prevenir inyecciones SQL.

BIBLIOGRAFIA:

Akhawe, D., & Lam, A. (2013). *Blind SQL Injection: A Complete Attack and Defense Guide*. SANS Institute.

https://brightsec.com/blog/error-based-sql-injection/
https://openwebinars.net/blog/que-es-sql-injection/#:~:text=Sql%20Injection%20%C3%B3%20Inyecci%C3%B3n%20SQL,datos%20almacenados%20estar%C3%ADan%20en%20peligro.
https://powerdmarc.com/es/sql-injection-prevention/