TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TLAXIACO

SEGURIDAD Y VIRTUALIZACIÓN

Investigación Practica 4 - Inyección SQL

Integrantes del Equipo:

Arnol Jesus Cruz Ortiz

Amilkar Vladimir Reyes Reyes

Rael Gabriel Bautista

Sandra Gabriela Velasco Guzmán

Docente:

Edward Osorio Salinas

Materia:

Seguridad y virtualización

Carrera:

Ingeniera en Sistemas Computacionales

Grupo: 7US

Semestre: Agosto – diciembre 2024

26/Septiembre/2024

ÍNDICE

¿QUÉ ES LA INYECCIÓN DE SQL?

La inyección de SQL es un tipo de ciberataque encubierto en el cual un hacker inserta código propio en un sitio web con el fin de quebrantar las medidas de seguridad y acceder a datos protegidos. Una vez dentro, puede controlar la base de datos del sitio web y secuestrar la información de los usuarios. Le explicamos cómo funcionan los ataques de inyección de SQL, cómo combatirlos y cómo una herramienta antivirus potente lo puede proteger contra las consecuencias.

¿Y dónde entra la parte de la inyección?

Si un desarrollador web no es meticuloso, al crear un sitio podría dejar un resquicio que alguien con malas intenciones podría usar para provocar efectos inesperados en su base de datos. Las inyecciones de SQL (o SQLI) se producen cuando el hacker introduce o inyecta en el sitio web código SQL malicioso, un tipo de malware que se conoce como la carga útil, y consigue subrepticiamente que envíe ese código a su base de datos como si de una consulta legítima se tratara.

BLIND SQL INJECTION (SQLI)

Es una técnica de ataque en la que el atacante no puede ver directamente los resultados de las consultas SQL inyectadas, pero puede inferir el estado de la base de datos en función de las respuestas indirectas de la aplicación. Este tipo de ataque es utilizado cuando una aplicación no devuelve errores detallados ni respuestas visibles, pero sigue siendo vulnerable a inyecciones SQL.

Existen dos tipos principales de Blind SQL Injection:

1. Boolean-Based Blind SQL Injection

En este tipo de ataque, el atacante utiliza consultas SQL que devuelven una respuesta booleana (verdadera o falsa). Según la respuesta que obtenga (por ejemplo, si la página carga correctamente o no, o si redirige a una página diferente), el atacante puede deducir si la inyección fue exitosa y extraer información de la base de datos. Este método es más lento ya que se requiere probar muchas condiciones booleanas para extraer datos.

2. Time-Based Blind SQL Injection

En la inyección SQL basada en tiempo, el atacante utiliza comandos SQL que retrasan intencionalmente la respuesta del servidor para determinar si una inyección es exitosa. Por ejemplo, el atacante puede inyectar una consulta que hace que el sistema espere un número de segundos antes de responder. Si la página tarda más en cargarse después de la inyección, esto indica que la consulta fue ejecutada.

Protección contra Blind SQL Injection:

Consultas preparadas y consultas parametrizadas: Utilizar siempre consultas parametrizadas para evitar la manipulación de las entradas de usuario.

Validación de entrada: Verificar y sanitizar las entradas de usuario para asegurar que solo los datos esperados lleguen a la base de datos.

Mensajes de error genéricos: No proporcionar mensajes de error detallados que puedan ofrecer información sobre la estructura de la base de datos.

Herramientas para detectar Blind SQL Injection:

SQLMap: Es una herramienta de código abierto que puede detectar y explotar diferentes tipos de inyecciones SQL, incluyendo las Blind SQL Injection.

Burp Suite: Utilizada para pruebas de seguridad web, también puede identificar este tipo de vulnerabilidades.

SQL INJECTION BASADA EN ERRORES

Es una técnica de inyección SQL en la que el atacante explota los mensajes de error que devuelve la base de datos para obtener información sobre la estructura de la misma. Estos mensajes detallados proporcionados por el servidor pueden revelar información sensible, como nombres de tablas, columnas, e incluso datos contenidos en la base de datos.

¿Cómo funciona?

Este tipo de inyección SQL depende de la visibilidad de los errores que la base de datos genera cuando se ejecutan consultas incorrectas. Los sistemas mal configurados, que permiten que los errores sean visibles para el usuario, son particularmente vulnerables a este ataque. Los atacantes inyectan comandos

SQL maliciosos que están diseñados para provocar errores deliberadamente. Al observar los mensajes de error, pueden deducir cómo está estructurada la base de datos y qué datos están disponibles.

Tipos de errores comunes explotados:

Errores de sintaxis SQL: Cuando se envían consultas mal formadas, la base de datos puede devolver detalles sobre la sintaxis esperada.

Errores de tipo de datos: Cuando se esperan tipos de datos específicos (como enteros) y se proporcionan otros tipos (como cadenas), la base de datos puede indicar la estructura de las tablas o las columnas.

Errores de permisos: Algunas veces, las bases de datos mal configuradas pueden devolver errores de permisos que revelan los nombres de las tablas o incluso qué usuario de la base de datos está ejecutando las consultas.

Consecuencias de SQL Injection basada en errores:

Reconocimiento de la estructura de la base de datos: Los atacantes pueden conocer qué tablas y columnas existen, lo que facilita la extracción de datos sensibles.

Exfiltración de datos: Dependiendo de los mensajes de error, los atacantes podrían extraer directamente información como nombres de usuarios, contraseñas, o correos electrónicos.

Escalamiento de ataques: Los errores expuestos pueden proporcionar las pistas necesarias para que un atacante pase a formas más complejas de inyección SQL, como la basada en tiempo o ciega.

Prevención de SQL Injection basada en errores:

Validación de entradas: Siempre validar y sanitizar las entradas del usuario, usando consultas parametrizadas o procedimientos almacenados.

Deshabilitar mensajes de error detallados: Configurar el servidor para que no devuelva detalles de los errores SQL al cliente, usando logs internos para la depuración.

Usar consultas preparadas: Evitar construir consultas SQL dinámicamente con las entradas de usuario y usar consultas preparadas o ORMs (Object-Relational Mappers) que previenen este tipo de inyección.

Herramientas de detección:

SQLMap: Una herramienta automatizada de código abierto que puede detectar múltiples formas de inyección SQL, incluyendo las basadas en errores.

Acunetix: Un escáner de seguridad web que detecta y reporta vulnerabilidades como inyecciones SQL.

LA SQL INJECTION BASADA EN TIEMPO

Es una variante de la inyección SQL ciega que se utiliza cuando no se pueden ver directamente los resultados de las consultas SQL inyectadas. A diferencia de otras formas de inyección SQL, esta técnica no se basa en los mensajes de error ni en la visualización de datos, sino en medir el tiempo que tarda en responder el servidor ante una consulta inyectada.

¿Cómo funciona?

En este tipo de ataque, el atacante inserta consultas que provocan un retraso en la respuesta del servidor si una determinada condición es verdadera. Si la página tarda en responder, el atacante puede inferir que la consulta se ejecutó correctamente. Este proceso es más lento que otros tipos de inyección SQL, pero es eficaz cuando la aplicación no muestra errores ni resultados visibles.

Prevención:

Consultas preparadas y parametrizadas: Las consultas preparadas evitan que las entradas de usuario alteren la lógica de las consultas SQL.

Escapar correctamente las entradas: Asegurarse de que todos los datos proporcionados por el usuario se validen y escapen correctamente antes de usarlos en una consulta SQL.

Limitación de funciones de retraso: Restringir el uso de funciones que generen retrasos en las consultas SQL para minimizar la posibilidad de ataques de este tipo.

Herramientas para detectar inyección basada en tiempo:

SQLMap: Un poderoso detector de vulnerabilidades SQL que puede identificar Time-Based Blind SQL Injection y otros tipos de inyección.

Burp Suite: Con capacidades avanzadas para detectar esta técnica mediante la medición del tiempo de respuesta en las solicitudes HTTP.

LA SQL INJECTION EN PROCEDIMIENTOS ALMACENADOS

Ocurre cuando un atacante explota vulnerabilidades en los procedimientos almacenados (o stored procedures) de una base de datos. Un procedimiento almacenado es un conjunto predefinido de instrucciones SQL que se pueden reutilizar. Si estos procedimientos no están diseñados correctamente, pueden ser vulnerables a ataques de inyección SQL, lo que permite que un atacante manipule consultas para acceder, modificar o eliminar datos no autorizados.

¿Cómo ocurre la SQL Injection en procedimientos almacenados?

Los procedimientos almacenados pueden ser vulnerables si:

Utilizan concatenación de cadenas para construir consultas SQL dinámicas a partir de la entrada del usuario.

No validan adecuadamente los parámetros de entrada, lo que permite que un atacante inserte código SQL malicioso.

Impacto de la inyección SQL en procedimientos almacenados:

Acceso no autorizado: Un atacante puede obtener acceso a datos sensibles que no debería ver, como credenciales o información personal.

Manipulación de datos: Puede alterar o eliminar registros importantes en la base de datos.

Control total del servidor: En casos más graves, el atacante puede ejecutar comandos en el servidor de base de datos, comprometiendo todo el sistema.

Tipos de ataques en procedimientos almacenados:

Inyección directa: Cuando los datos del usuario se concatenan directamente en la consulta sin validación.

Inyección indirecta: Cuando el procedimiento almacenado llama a otro procedimiento con datos no validados.

SQL INJECTION EN ORM

Ocurre cuando las consultas a la base de datos, generadas a través de un ORM, no están adecuadamente protegidas contra entradas maliciosas. Aunque los ORMs abstraen la interacción directa con SQL y generalmente ofrecen protección contra inyecciones SQL, aún pueden ser vulnerables cuando se utilizan consultas dinámicas mal gestionadas.

¿Cómo funciona?

Los ORMs permiten a los desarrolladores interactuar con bases de datos usando el paradigma de programación orientado a objetos en lugar de escribir consultas SQL manualmente. Sin embargo, algunos ORMs permiten generar consultas SQL dinámicas dentro del código, lo que puede generar riesgos de inyección SQL si no se validan correctamente los datos de entrada

HERRAMIENTAS PARA DETECTAR Y PREVENIR SQL INJECTION

Para detectar y prevenir SQL Injection, se pueden utilizar diversas herramientas que van desde escáneres automáticos de vulnerabilidades hasta soluciones de seguridad integradas en el ciclo de desarrollo. Aquí están algunas de las principales herramientas para cada propósito:

1. Herramientas para Detectar SQL Injection:

Estas herramientas permiten identificar las vulnerabilidades de SQL Injection en una aplicación, ya sea mediante análisis estático (del código) o pruebas dinámicas (de la aplicación en ejecución).

SQLMap: Es una herramienta de código abierto ampliamente utilizada para detectar y explotar vulnerabilidades de SQL Injection. Ofrece una interfaz de línea de comandos y permite ejecutar una amplia gama de ataques SQL Injection, desde inyecciones simples hasta avanzadas (time-based, error-based, union-based, etc.).

Burp Suite: Este conjunto de herramientas de pentesting tiene un módulo de escaneo automático que puede identificar vulnerabilidades de SQL Injection.

Además, con su función de proxy, permite la inspección manual de las solicitudes y la inyección de comandos maliciosos para verificar vulnerabilidades.

Acunetix: Escáner automatizado de aplicaciones web que busca diversas vulnerabilidades, incluidas las inyecciones SQL. También ofrece análisis de código y pruebas basadas en comportamiento.

Netsparker: Este escáner automático es conocido por identificar con precisión vulnerabilidades de SQL Injection y otros problemas de seguridad en aplicaciones web, utilizando tanto análisis dinámico como estático.

OWASP ZAP (Zed Attack Proxy): Una herramienta de seguridad gratuita y de código abierto, desarrollada por OWASP, que ayuda a encontrar vulnerabilidades en aplicaciones web, incluidas inyecciones SQL. Se puede usar tanto manualmente como con su escáner automatizado.

2. Herramientas para Prevenir SQL Injection:

Además de la detección, la prevención es clave para proteger una aplicación de inyecciones SQL. Aquí hay algunas herramientas y enfoques que ayudan a prevenir SQL Injection:

Firewalls de Aplicación Web (WAFs): Un WAF monitorea, filtra y bloquea tráfico HTTP malicioso. Algunos WAFs pueden detectar patrones de inyección SQL y bloquear solicitudes sospechosas antes de que lleguen al servidor. Ejemplos de WAFs populares incluyen:

ModSecurity: Un WAF de código abierto que puede integrarse en servidores web como Apache o Nginx.

Cloudflare: Ofrece protección WAF en la nube que incluye reglas para bloquear inyecciones SQL.

ORM (Object-Relational Mapping): Utilizar ORMs como Entity Framework, Django ORM o Hibernate proporciona una capa de abstracción sobre SQL que reduce el riesgo de inyecciones SQL al manejar automáticamente las consultas a la base de datos de forma segura. Los ORMs generalmente usan consultas parametrizadas, lo que evita la inyección directa de código SQL malicioso.

Escapado y validación de entradas: Herramientas y bibliotecas como OWASP ESAPI (Enterprise Security API) permiten un manejo más seguro de las entradas de usuario al escaparlas correctamente para prevenir inyecciones SQL.

Preparar declaraciones SQL: La mayoría de los lenguajes y frameworks modernos ofrecen soporte para consultas parametrizadas. Herramientas como Prepared Statements o Parameterized Queries aseguran que las entradas de usuario se manejen de manera segura.

3. Otras Herramientas Útiles:

SonarQube: Un analizador de código que detecta vulnerabilidades en el código fuente, incluidas inyecciones SQL. Ofrece soporte para muchos lenguajes de programación y se integra fácilmente en entornos de CI/CD (Integración continua/Entrega continua).

Veracode: Proporciona análisis estático de aplicaciones para detectar vulnerabilidades de seguridad, como SQL Injection, en el código fuente o binario.

CONCLUSIÓN

Las herramientas más efectivas para proteger contra inyecciones SQL son aquellas que detectan y previenen a lo largo de todo el ciclo de desarrollo de la aplicación, como análisis de código, escáneres automáticos y WAFs. Es esencial utilizar un enfoque en capas para maximizar la protección, incluyendo consultas parametrizadas, validación de entradas, y monitorización activa del tráfico.

Bibliografias

Blind SQL Injection | OWASP Foundation. (s. f.). https://owasp.org/www-community/attacks/Blind_SQL_Injection

Belcic, I. (2023, 23 febrero). *¿Qué es la inyección de SQL y cómo funciona?* ¿Qué Es la Inyección de SQL y Cómo Funciona? https://www.avast.com/es-es/c-sql-injection