Principales enfoques y métodos para la detección de iSQL

Stanislav Glebskiy
Instituto Nacional de Astrofísica Óptica y Electrónica
Puebla, México
stanislav.glebskiy@inaoep.mx

Dra. Lil María Xibai Rodríguez Henríquez,
Instituto Nacional de Astrofísica Óptica y Electrónica
Puebla, México
lmrodriguez@inaoep.mx

***Abstract*—Las inyecciones de SQL son un tema de gran relevancia en la protección de datos digitales almacenados en bases de datos relacionales. En este artículo se habla de manera general sobre las tendencias en el desarrollo de métodos para detectar y prevenir este tipo de ataques, enfocándose tanto en el sector industrial como en el ámbito de investigación académica. Se presentan de forma simplificada los conceptos necesarios para entender el tema. Este artículo pretende servir de introducción a cualquier persona que quiera adentrarse en el mundo de las iSQL (inyecciones SQL) y su mitigación, presentando un panorama general del área y discutiendo los enfoques más recientes de investigación.**

***Palabras clave*—Inyecciones SQL, bases de datos, seguridad.**

I. Introducción

Gran parte de la información que se maneja en la sociedad se almacena hoy en día de manera digital. Un método de almacenamiento de información digital son las Bases de Datos Relacionales. Estas son implementadas a través de Sistemas de Manejo de Bases de Datos Relacionales o RDBMS (por sus siglas en inglés [13]). El mantenimiento, uso y gestión adecuados de las bases de datos son relevantes para las empresas e instituciones, ya que les permiten tomar decisiones informadas y mantener una operación eficiente al salvaguardar los datos de la empresa de manera segura, disponible e íntegra.

Cualquier persona, institución o empresa que pretenda mantener una base de datos debe garantizar la seguridad de la información que contiene. Existen retos de seguridad específicos que conciernen a las Bases de Datos Relacionales. Uno de los retos más destacados es la detección y prevención de iSQL. Los ataques de inyección de SQL son una de las vulnerabilidades más comunes y peligrosas que afligen a los sistemas que contienen bases de datos. Según la OWASP, estos ataques figuraban como el ataque más común en el año 2017 y el tercero más común en el año 2021 [1]. Utilizando iSQL, un usuario malintencionado puede acceder a información confidencial o incluso llegar a alterar la información de la base de datos.

En este artículo se pretende dar una introducción simplificada al tema de las inyecciones SQL y las diferentes maneras de mitigarlas. Se introducen los conceptos esenciales necesarios para entender las iSQL como el entorno en el que ocurren, cómo se definen y cómo se clasifican. Posteriormente, se procede a exponer las soluciones que se han propuesto para mitigar estas vulnerabilidades, distinguiendo dos ramas principales: soluciones que se ven en entornos industriales y soluciones que están siendo desarrolladas en el ámbito académico. Para ilustrar los ejemplos presentados más adelante se toma como referencia una base de datos denominada ”empresa” presentada en la Figura 1 que está compuesta por dos tablas: ”almacén” y ”empleados”.

Fig. 1. Base de datos ”empresa”

En la sección II se explica qué es una inyección SQL, en la sección III se mencionan las principales categorías de iSQL que existen y se presentan algunos ejemplos, en la sección IV se habla del entorno donde se producen las iSQL y donde se pueden mitigar, en la sección V se habla de los principales enfoques de mitigación de iSQL utilizados en sistemas de producción, mientras que en la sección VI se mencionan los enfoques que se pueden encontrar en el mundo académico, en la sección VII se presenta la discusión donde se comparan los enfoques industriales y académicos y se detallan las tendencias actuales del campo, por último, en la sección VIII se muestran las conclusiones del artículo.

II. ¿Qué es una iSQL?

SQL (del inglés Structured Query Language) es el lenguaje que se utiliza para interactuar con las bases de datos. Tanto aplicaciones como personas pueden utilizar SQL. SQL son los comandos que un RDBMS puede entender y ejecutar. Por lo general, las aplicaciones web utilizan datos

proporcionados por los usuarios para formular comandos SQL.

Una iSQL se puede definir como la alteración de un comando SQL para lograr un comportamiento de la base de datos diferente al normal mediante el uso de entradas de usuario que se concatenan al comando SQL [13].

Se propone un ejemplo donde se tiene una aplicación web que se dedica a la venta de artículos por internet. Para llevar la gestión de todos sus productos, ventas, usuarios, etc., la aplicación web utiliza la base de datos relacional ”empresa” de la Figura 1. Cuando un cliente hace una consulta, selecciona el tipo de artículos que le interesa de una lista disponible en la página web. La página web recibe esta información en forma de una cadena de caracteres y genera un comando SQL para que el RDBMS le entregue todos los artículos relevantes que existen en la base de datos. En la Figura 2 se muestra este sistema en funcionamiento con un cliente seleccionando la categoría de ”ropa”. Así se vería un comando SQL benigno generado en la situación descrita en el ejemplo anterior:

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa';
```

Pero el hecho de que la aplicación web maneje las categorías como cadenas de caracteres y las concatene directamente en el comando SQL abre la puerta a las iSQL. Si los usuarios proporcionan palabras o caracteres que tienen un significado en el lenguaje SQL, la función del comando resultante puede ser alterada. Un comando SQL alterado por una iSQL se vería como:

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa' or 1=1 ;-- ';
```

En este caso el usuario logró modificar la información que se envía al servidor web para que la categoría fuera “ropa’ or 1=1 ;– ”, en vez de ”ropa”. Esta entrada alteraría la manera en la que el RDBMS procesa el comando SQL. En el primer caso el RDBMS devolvería todos los productos de la categoría ”ropa”, pero en el segundo caso expondría todos los productos en la tabla ”almacen”. Esto podría no ser deseado si la tabla contiene categorías de productos que no deberían estar expuestas a los clientes.

Aunque las iSQLs se conocen desde mediados de los años 90s, hasta el día de hoy, es un tema que no se ha resuelto en su totalidad. Se han propuesto muchas soluciones, pero es difícil encontrar una que abarque todos los tipos de iSQL y ambientes de implementación, ya que como se verá en la Sección III, estos ataques son muy variados.

La razón por la que es difícil prevenir las iSQL es porque, de acuerdo con Su Z. et. al. [2] existe una brecha semántica entre la aplicación web que genera el comando y la base de datos que interpreta el comando. Es decir, la aplicación web no entiende propiamente el comando SQL, para la aplicación web, el comando SQL solo es una cadena de caracteres.

III. Tipos de iSQLs

Existen varios tipos de iSQL. A continuación, mencionamos los tipos de iSQL más conocidos categorizados por cómo se altera el comando SQL propuesto por Devi R. et. al. en [19] .

A. Tautologías

Se basan en la cláusula WHERE del lenguaje SQL. Son aquellas donde se inyecta una condición que siempre es cierta en la cláusula WHERE de un comando SELECT. De esta forma, se logra invalidar los filtros de una búsqueda SQL y obtener información restringida. En la sección II, se mostró un ejemplo de un iSQL de este tipo. Esta inyección le permitiría a un atacante obtener todo el contenido de la tabla ”almacén” de la Figura 1.

B. Unión

Son aquellas donde inyectando la cláusula UNION en un comando SELECT el atacante es capaz de obtener información de tablas o columnas adicionales. A continuación se muestra un ejemplo de esta iSQL:

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa' UNION
  SELECT nombre
  FROM empleados;-- ';
```

Esta inyección le permitiría a un atacante acceder al contenido de la tabla ”empleados” de la Figura 1.

C. Piggyback query

Son aquellas donde, mediante la inyección de caracteres de fin de comando, como “;”, el atacante es capaz de agregar un segundo comando SQL para que este sea ejecutado junto al comando original. A continuación se muestra un ejemplo de esta iSQL:

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa';
  DROP DATABASE empresa;-- ';
```

D. Basadas en errores

Son aquellas donde se extrae información del sistema de base de datos al provocar la ejecución de un comando SQL mal formado. La base de datos retorna un mensaje de error y el atacante puede obtener información del sistema a partir de este.

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa' WHERE;-- ';
```

De igual manera, de acuerdo a Rai A. et. al. en [18], las iSQL se pueden categorizar por la manera en la que se ejecutan:

E. Clásicas

Son aquellas donde el atacante obtiene la respuesta de un intento de inyección por el mismo canal por donde se realiza el ataque. Este es el tipo de iSQL más común. Un ejemplo sería cuando un atacante manda inyecciones SQL a través de una página web y observa la respuesta del sistema en la misma página web. Todos los ejemplos de iSQL presentados anteriormente se pueden ejecutar de esta forma.

F. De segundo orden

Se dan cuando el atacante logra almacenar un comando SQL manipulado directamente en la base de datos. Este comando malicioso es ejecutado en el futuro. En este caso, la entrada maliciosa no proviene directamente del usuario, sino de la base de datos misma. Por ejemplo, el atacante logra agregar un nuevo producto a la tabla "almacén" de la base de datos de la Figura 1 con el siguiente comando SQL:

```
INSERT INTO almacen (id,
  producto,
  categoria,
  piezas)
VALUES (6,
  'producto6',
  'ropa''; DROP DATABASE empresa ',
  1);
```

Posteriormente, si algún programa lee la columna "categoría" de este nuevo producto y la concatena a otro comando SQL, puede provocar una iSQL de tipo "Piggyback query" y eliminar toda la base de datos.

G. Ciegas

Son aquellas donde el atacante no es capaz de visualizar una respuesta directa del RDBMS, pero puede manipular algún componente del sistema. El ejemplo más común es cuando un atacante logra ejecutar una cláusula SLEEP() en el RDBMS y, midiendo el tiempo de respuesta del sistema, puede saber que su ataque está siendo ejecutado en el RDBMS. A continuación se presenta un ejemplo:

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa' or
  SLEEP(10);-- ';
```

H. Out-of-bound

Son aquellas donde el canal por donde se envía el ataque es diferente al canal por donde se recibe la respuesta. Por ejemplo, el atacante inyecta un SQL malicioso en una página web que hace que el RDBMS guarde información sensible en un archivo del SO (sistema operativo). Posteriormente, el atacante accede por otro medio al SO y descarga el archivo, obteniendo así la información de la base de datos.

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa' or 1=1 INTO
  OUTFILE '/tmp/myfile';-- ';
```

IV. Entorno de iSQL

Cuando se hace referencia a las iSQL generalmente se considera el contexto de aplicaciones web. Si bien, en muchas ocasiones estas aplicaciones llegan a tener arquitecturas de hardware y software complejas, se pueden simplificar a una arquitectura de tres capas [13] que nos permite entender el flujo de datos y cómo se generan las peticiones SQL. En la Figura 2 podemos distinguir los elementos que conforman este modelo:

1) Los clientes
2) El servidor de la aplicación web
3) El servidor de la base de datos

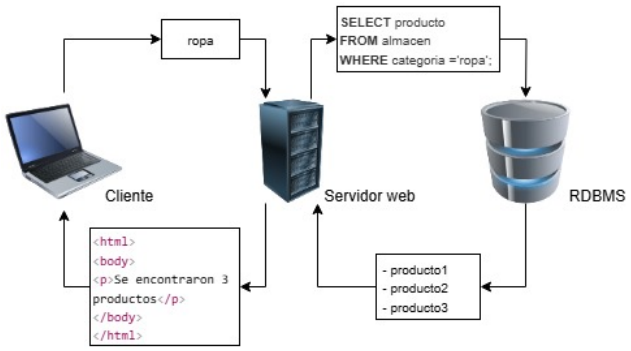

Fig. 2. Arquitectura web de tres capas.

En esta arquitectura, los comandos SQL son generados por la aplicación web (que reside en el servidor web) a partir de la información enviada por el cliente. El resultado del comando SQL es utilizado para devolver la información necesaria al cliente, por lo general como contenido de una página web. En la Figura 2 se observa el flujo de información, empezando por el cliente haciendo una petición que llega al servidor web. Posteriormente, la aplicación web hace una consulta a la base de datos utilizando lenguaje SQL. Por último, la información devuelta por la base de datos es utilizada para crear el contenido de la página web que se le devuelve al cliente.

Un sistema de detección de intrusión IDS (por sus siglas en inglés) de iSQL se puede implementar en cada uno de los elementos de una arquitectura de tres capas y también en los canales de comunicación entre estos elementos. Por lo que, de acuerdo a Kar D. et. al. [3], podemos definir 5 niveles en los que se puede implementar un IDS de iSQL:

1) En las aplicaciones de los clientes.
2) Entre los clientes y la aplicación web (firewall de la aplicación web).
3) En la aplicación web.
4) Entre la aplicación web y el RDBMS (firewall del RDBMS).
5) En el RDBMS.

Por lo general, una misma base de datos suele ser usada por varias aplicaciones web. Y una aplicación web suele tener varios clientes. Por lo que, si un IDS de iSQL se implementa en un nivel más bajo, se tiene que implementar en más instancias. Considerando el caso de una única base de datos que sirve a tres aplicaciones web distintas. Basta con que una sola aplicación web sea vulnerable a iSQL para que toda la base de datos quede comprometida.

Cuando se habla de soluciones a las iSQL se puede distinguir entre el sector industrial, donde se implementan soluciones maduras en ambientes de producción, o el sector académico, donde se proponen y estudian soluciones más innovadoras, pero menos maduras.

V. Soluciones industriales

Los IDS de iSQL que se implementan en ambientes de producción tienen que ser sistemas ampliamente probados y que no tengan efectos negativos como ralentizar el procesamiento de las bases de datos o aplicaciones web. Los principales enfoques para solucionar las iSQL en ambientes industriales son:

A. Expresiones regulares

Estos son mecanismos que detectan patrones comunes de iSQL mediante el uso de expresiones regulares. Estos mecanismos se encuentran comúnmente implementados en los firewalls a distintos niveles en la arquitectura web. Por lo general, los firewalls de red y de aplicación web (por sus siglas en inglés WAF) no han tenido un buen desempeño en la detección de ataques de iSQL [16]. Los firewalls de bases de datos suelen tener mejor desempeño en la detección de ataques de iSQL. Aun así, los sistemas IDS de iSQL basados en expresiones regulares no son buenos para detectar iSQL complejas y pueden ser engañados con técnicas de codificación del ataque iSQL [15].

B. Listas negras y blancas

Las listas negras son conjuntos de comandos SQL o expresiones que el sistema bloquea. Las listas blancas, por el contrario, son listas de comandos SQL que el sistema deja pasar, es decir, bloquea todos los comandos SQL a excepción de los que están en la lista blanca. Ejemplos de sistemas que utilizan esta técnica se pueden ver en [5] y [6].

C. Verificación de código de aplicaciones web

Una aplicación web que está correctamente diseñada y cuenta con una buena implementación, no deja pasar iSQL. Algunas de las recomendaciones que se deben seguir al diseñar aplicaciones web que trabajen con bases de datos [15] son:

- Validación de datos proporcionados por los usuarios.
- Uso de comandos SQL parametrizados.
- Uso de procedimientos de RDBMS almacenados.
- Codificación de los datos.

Sidik R. F. et. al. [17] demostró la eficacia del uso de comandos SQL parametrizados en el lenguaje Golang al mitigar todas las vulnerabilidades de iSQL en su ambiente de experimentación. Estos métodos se implementan mediante la capacitación de los desarrolladores web. De igual manera, existen analizadores de código que pueden detectar vulnerabilidades de iSQLs y sugerir algunos de los métodos presentados anteriormente para mejorar el código fuente.

D. Configuración segura del entorno

Por último, el seguimiento de normas y estándares de seguridad es la principal defensa contra las iSQL. Estos incluyen:

- Segmentación de los datos.
- Cifrado de datos sensibles.
- Manejo adecuando de usuarios.
- Principio del mínimo privilegió.

Estos se deben aplicar tanto durante el desarrollo de las aplicaciones y bases de datos como durante su ejecución en ambientes de producción. La configuración correcta de un sistema elimina gran parte de las posibles vulnerabilidades. Como ejemplo, Hamidi A. [7] explora los diferentes mecanismos que interactúan en el RDBMS de MySQL para proporcionar seguridad a la base de datos y habla sobre la correcta configuración de estos.

VI. Soluciones académicas

En el ámbito académico se encuentra una mayor variedad de enfoques y mecanismos para la mitigación de las iSQL. Cabe mencionar que todas las soluciones utilizadas en el sector industrial se siguen estudiando y mejorando en el mundo académico, pero también se pueden observar enfoques nuevos.

A. Comparación de árbol de parseo

Un enfoque interesante es la comparación de los árboles de parseo de un comando SQL antes y después del input de los usuarios propuesto por Buehrer G. et. al. en [8]. Cuando un comando SQL es procesado por un RDBMS pasa por varias etapas. Una de ellas es la transformación del comando SQL en un árbol donde cada elemento del comando es representado por un nodo. Diferentes comandos SQL van a resultar en diferentes árboles. Una misma estructura de un árbol puede

contener distintos datos. Es decir, que la estructura del árbol representa el esqueleto del comando SQL. Si un comando SQL es alterado, su funcionalidad cambia y va a producir un árbol diferente. Otro ejemplo de este enfoque es presentado por Katole R. et. al. en [9].

B. Métodos basados en tokens

Este enfoque, observado por Shahbaaz M en [24], se centra en expresar un comando SQL mediante una serie de tokens predefinidos. De esta forma, se logra transformar el comando original, eliminando información irrelevante y preservando las características esenciales del comando SQL. Analizar los tokens simplifica la identificación de patrones de iSQL. Los trabajos de Kar D. et. al. en [4] y [3] son buenas ejemplificaciones de este enfoque.

C. Métodos basados en comportamiento

Por otro lado, existe el enfoque basado en detectar comportamiento anómalo para prevenir ataques de iSQL. Estos métodos se basan en establecer un modelo de lo que es "normal" para un sistema y, cuando detectan que algo sale de ese patrón, lo clasifican como peligroso. Estos métodos son similares en su esencia a los sistemas UEBA (del sus siglas en inglés User and Entity Behavior Analytics), donde se modela el comportamiento de usuario y se determina si una acción es "normal" o "anormal".

Esta es una gama bastante amplia de soluciones, ya que se pueden implementar de distintas maneras y en distintos niveles de los sistemas web. Medeiros I. et. al. [10] presenta un sistema implementado dentro del mismo RDBMS, mientras que Fonseca J. et. al. [11] presenta un mecanismo implementado como Firewall del RDBMS. Algo característico de este enfoque es que necesita de una fase de aprendizaje. Es decir, existe un intervalo de tiempo donde el sistema debe aprender lo que se clasifica como peligroso y lo que no lo es. El aprendizaje se puede hacer con distintos métodos, pero sobresalen dos categorías más comunes:

- El aprendizaje se hace durante el funcionamiento normal del sistema. En este caso el sistema aprende lo que es el funcionamiento "normal" y si posteriormente detecta algo que no haya visto, lo marca como "anormal".
- Se cuenta con un conjunto de entradas peligrosas y no peligrosas. En este caso el administrador es encargado alimentar el sistema con las entradas ya marcadas.

Por lo general, las soluciones basadas en comportamiento son dependientes de su entorno de implementación. Es decir, su funcionamiento depende del ambiente en el que se estén ejecutando. Esto tiene sentido, ya que los ataques de iSQL se ajustan para cada sistema específico. Lo que puede ser una iSQL en un entorno puede no serlo en otro.

D. Machine Learning e inteligencia artificial

El auge de las tecnologías de Machine Learning e Inteligencia Artificial ha impulsado mucho la rama de IDS de iSQL [12]. Mientras que la idea general de este enfoque es similar a las soluciones basadas en comportamiento: aprender a diferenciar entre entradas "normales" y "anormales" a partir de un conjunto de datos, ahora se cuenta con herramientas matemáticas mucho más potentes para lograr este objetivo. En los últimos años se ha observado un gran número de trabajos académicos donde se han probado distintos modelos y algoritmos de Machine Learning para la detección de iSQL. Lu Z. [21] propone un clasificador Naive Bayes para detectar iSQL, Shahbaz M. et. al. [22] proponen un sistema basado en redes neuronales convolucionales que logra una precisión de 97%. Por otro lado, Shakya et. al. [23] en su trabajo implementaron sistemas de detección de iSQL usando algoritmos de SVM (del inglés Support Vector Machine), KNN (de inglés K-Nearest Neighbor) y RF (del inglés Random Forest) logrando precisiones de 98.28%, 99.55% y 87.72% respectivamente.

Maha et. al. [14] presenta la comparación de 36 propuestas de sistemas de detección de iSQL basados en Machine Learning publicados entre los años 2012 y 2021. Frecuentemente, los algoritmos de Machine Learning se usan para mejorar soluciones ya existentes. Babaey V. et. al. [20] propone un sistema basado en inteligencia artificial para generar reglas de WAF más eficientes.

VII. Discusión

Se observa una diferencia significativa entre las soluciones industriales y las propuestas académicas en el ámbito de detección de iSQL. Los sistemas de producción suelen utilizar métodos más básicos, pero bien probados y pulidos a través de varias décadas. Estos sistemas suelen cubrir los intentos de ataques de iSQL más comunes. En la Sección V se presentaron los enfoques de uso de expresiones regulares, listas negras y blancas, verificación de código de aplicaciones web y configuración segura del entorno. La verificación de código de la aplicación web y la configuración segura del entorno son la primera línea de defensa contra los ataques de iSQL y los que más intentos de iSQL previenen. Esto se debe a que, a diferencia de los otros dos métodos, previenen los ataques desde antes de que ocurran, mientras que los demás métodos mitigan el ataque cuando ya está en ejecución.

Los enfoques académicos modernos, si bien, muestran resultados prometedores, suelen ser soluciones bastante complejas y costosas. Surge la pregunta, ¿cuánto está dispuesto a invertir una empresa en un sistema IDS de iSQL? Si un sistema IDS de iSQL es demasiado caro o difícil de operar, muchas empresas no tendrán la posibilidad de implementarlo.

En la Sección VI se expusieron los enfoques de comparación de árbol de parseo, métodos basados en

tokens, métodos basados en comportamiento y los métodos basados en Machine Learning e inteligencia artificial, este último siendo una rama que se está desarrollando muy activamente en el ámbito académico. Este enfoque es bastante prometedor ya que, de acuerdo a Maha et. al. [14], las propuestas de IDS de iSQL que utilizan modelos de Machine Learning por lo general alcanzan porcentajes de detección arriba del 90%. Pero estas soluciones suelen ser computacionalmente costosas y requieren una etapa de aprendizaje. Aparte, esta etapa de aprendizaje suele atar el sistema de IDS de iSQL a un entorno específico, es decir, si se requiere modificar el entorno de ejecución del sistema IDS, se debe repetir el proceso de aprendizaje.

La tendencia más actual de la investigación académica está enfocada en identificar cómo se pueden utilizar los avances que se han hecho en el campo de la inteligencia artificial y Machine Learning en la detección y prevención de ataques iSQL. Se están probando distintos algoritmos de Machine Learning y redes neuronales para determinar cuáles son los mejores para identificar iSQL.

La solución más efectiva continúa siendo un diseño y programación seguros de aplicaciones web. Esto debido a que siempre es más fácil evitar un ataque que identificarlo y detenerlo en el momento de su ejecución. El problema de este enfoque es que da paso al error humano durante el proceso de desarrollo de aplicaciones web. Otra desventaja es que no siempre se tiene acceso al código fuente vulnerable. Aplicar estos métodos no es posible en software de terceros. Por lo que los sistemas IDS de iSQL siguen siendo necesarios.

VIII. Conclusión

Aún no se ha propuesto una solución definitiva a las iSQL. Existen varios enfoques para tratar de solucionar este problema. Por lo general, los métodos más probados y maduros se utilizan en la industria en sistemas de producción, mientras que el mundo académico está buscando enfoques más innovadores para atacar las iSQL. Las tendencias hacia la IA han marcado su huella en el diseño de nuevas soluciones a las iSQL. La detección y prevención de iSQL es un campo de investigación maduro, pero aún no agotado, ya que se están proponiendo nuevos enfoques para combatir la vulnerabilidad de las iSQL.

References

[1] OWASP, Owasp top ten, https://owasp.org/www-project-top-ten/, Visto: 2024-05-10.

[2] Zhendong Su, Gary Wassermann, "The essence of command injection attacks in web applications", 2006, http://dx.doi.org/10.1145/1111320.1111070.

[3] Debabrata Kar, Suvasini Panigrahi, Srikanth Sundararajan, "SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM", Computers & Security, Volumen 60, 2016, https://doi.org/10.1016/j.cose.2016.04.005.

[4] Debabrata Kar, Suvasini Panigrahi, Srikanth Sundararajan, "SQLiDDS: SQL Injection Detection using Query Transformation and Document Similarity", 11th International Conference on Distributed Computing and Internet Technology, 2015, http://dx.doi.org/10.1007/978-3-319-14977-6_41

[5] Oracle, Oracle SQL Firewall User's Guide, 2025, https://docs.oracle.com/en/database/oracle/oracle-database/23/sqlfw/overview-oracle-sql-firewall.html

[6] Oracle, MySQL Enterprise Firewall, 2024, https://www.mysql.com/products/enterprise/firewall.html

[7] Abdullah Hamidi, Abdul Razzaq Hamraz y Khadija Rahmani, "Database Security Mechanisms in MySQL", Afghanistan Research Journal - Natural Science, Volumen 4, 2022

[8] Gregory T. Buehrer, Bruce W. Weide y Paolo A. G. Sivilotti, "Using Parse Tree Validation to Prevent SQL Injection Attacks", Proceedings of the 5th International Workshop on Software Engineering and Middleware, 2005

[9] Rajashree A. Katole, Swati S. Sherekar y Vilas M. Thakare, "Detection of SQL Injection Attacks by Removing the Parameter Values of SQL Query", Proceedings of the Second International Conference on Inventive Systems and Control, 2018.

[10] Ibéria Medeiros, Beatriz Miguel, Nuno Neves y Miguel Correia, "Hacking the DBMS to Prevent Injection Attacks", 2016

[11] José Fonseca, Marco Vieira y Henrique Madeira, "Detecting Malicious SQL", TrustBus, Volumne 4657, 2007

[12] Mohammed Nasereddin, Malik Qasaimeha, Ashaar Alkhamaiseh y Raad Al-Qassas, "A systematic review of detection and prevention techniques of SQL injection attacks", Information Security Journal: A Global Perspective, Volumen 3548, 2021

[13] Ramez Elmasri y Shamkant B. Navathe, "Fundamentals of Database Systems", Pearson, 2016

[14] Maha Alghawazi, Daniyal Alghazzawi y Suaad Alarifi, "Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review", Journal of Cybersecurity and Privacy

[15] Justin Clarke, "SQL Injection Attacks and Defense", Elsevier, 2009

[16] Oracle, "Mitigating Risks of SQL Injection", 2023

[17] Rizaldi Fatah Sidik, Syifa Nurgaida Yutia y Rana Zaini Fathiyana, "The Effectiveness of Parameterized Queries in Preventing SQL Injection Attacks at Go", Proceedings of the International Conference on Enterprise and Industrial Systems, 2023

[18] Aditya Rai, Mazharul Islam Miraz, Deshbandhu Das, Harpreet Kaur y Swati, "SQL Injection: Classification and Prevention", International Conference on Intelligent Engineering and Management, 2021

[19] Rubidha Devi, Raghuraman Koteeswaran y Ramasamy Venkatesan, "A study on SQL injection techniques", International Journal of Pharmacy and Technology, 2016

[20] Vahid Babaey y Arun Ravindran, "GenSQLi: A Generative Artificial Intelligence Framework for Automatically Securing Web Application Firewalls Against Structured Query Language Injection Attacks", Future Internet, 2025

[21] Zhexi Lu, "Sql injection detection using Naïve Bayes classifier: a probabilistic approach for web application security", ITM Web of Conferences, 2025, https://doi.org/10.1051/itmconf/20257004016

[22] Muhammad Shahbaz, Gohar Mumtaz, Saleem Zubair y Mudassar Rehman, "Evaluating CNN Effectiveness in SQL Injection Attack Detection", Journal of Computing & Biomedical Informatics, 2024, https://doi.org/10.56979/702/2024

[23] Shakya, Dharmaratne y Manjula Sandirigama, "Detection of SQL Injection Attacks Using Machine Learning Techniques", 2024 International Conference on Electrical, Communication and Computer Engineering (ICECCE), 2024, https://doi.org/10.1109/ICECCE63537.2024.10823462

[24] Shahbaaz Mohammed Hayat Chaki, "A Comparative Analysis of SQL Injection Prevention Methods: Evaluating SQLPMDS vs SIUQAPTT Effectiveness", TechRxiv, 2025