TECNOLÓGICO NACIONAL DE MÉXICO.

INSTITUTO TECNOLÓGICO DE TLAXIACO.

INGENIERÍA EN SISTEMAS COMPUTACIONALES.

SEGURIDAD Y VIRTUALIZACION.

PRACTICA 4: INYECCIÓN SQL

INTEGRANTES DEL EQUIPO:

JEANETTE ARLET SALAZAR NICOLÁS.	21620202
NELSY ORTIZ LÓPEZ.	21620165
MARIBEL LUCERO ZUÑIGA.	21620139

SEMESTRE: SEPTIMO **GRUPO:** 7 US

ASESOR: EDWARD OSORIO SALINAS.

TLAXIACO, OAX, A 03 DE OCTUBRE DE 2024.

ÍNDICE DE ILUSTRACIONES

DESARROLLO DE LA PRÁCTICA.

1. **Crear una base de datos con una tabla que contenga al menos 3 registros.**

Ilustración 1: Tabla productos.

		id	nombre	descripcion	precio	cantidad
☐	Editar Copiar Borrar	1	Laptop	Laptop de alto rendimiento para trabajo y juegos	1200.00	0
☐	Editar Copiar Borrar	2	Mouse	Mouse inalámbrico con sensor óptico	20.00	0
☐	Editar Copiar Borrar	3	Teclado	Teclado mecánico con retroiluminación RGB	50.00	0

Ilustración 2: Registros en la tabla productos.

2. **Crear una aplicación web que permita buscar un registro por su id, nombre o descripción.**

A continuación, se presenta una aplicación sencilla que permite buscar registros utilizando su ID, nombre o descripción. Al ingresar datos en cualquiera de estos campos, el sistema mostrará en pantalla una tabla con los registros que coinciden con los criterios de búsqueda.

Ilustración 3: Interfaz de Aplicación que permita buscar registros.

Ilustración 4: Consulta de registros a través del ID.

3. Realizar pruebas de inyección de código en la aplicación web.

Ejemplo 1.

- Prueba básica: Inserta una coma simple (‘) en el campo de entrada.
- Ejemplo: '

Ilustración 5: Inyección de error SQL.

Ejemplo 2.

- Inyección básica: Payload sencillo para alterar una consulta.
- URL: http://localhost/buscar_producto.php?id=1 OR 1=1
- Objetivo: Si la consulta devuelve todos los registros de la tabla, la aplicación es vulnerable.

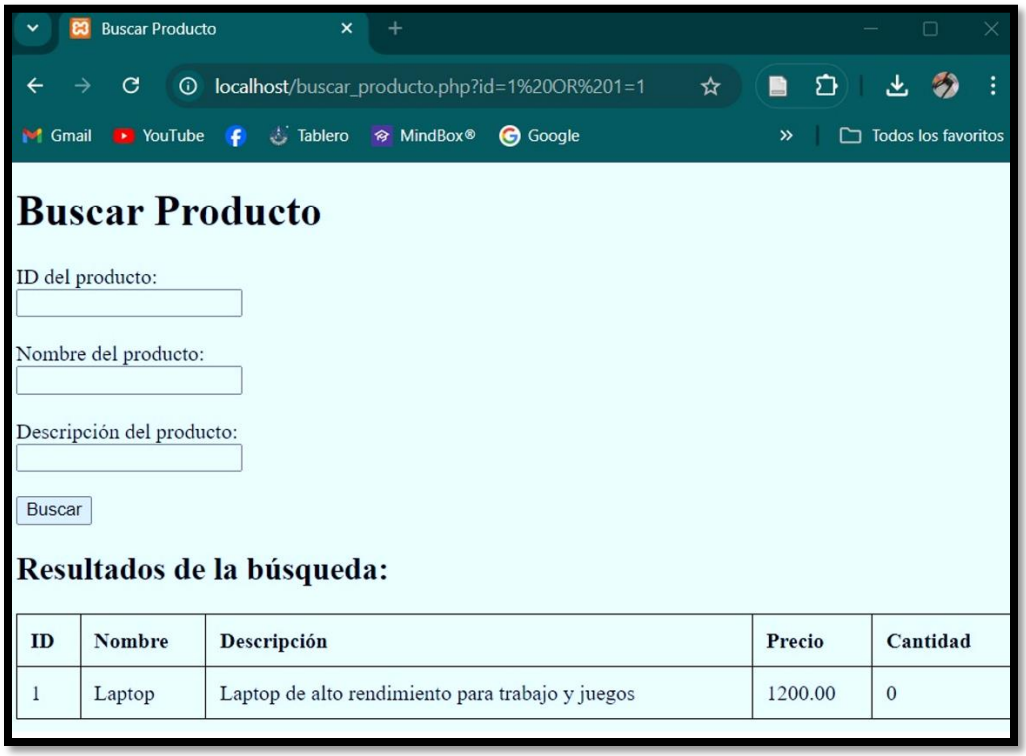

Ilustración 6: Inyección SQL de lógica booleana.

- Explotación de error: Inyección que devuelva un mensaje de error.
- URL: http://localhost/buscar_producto.php?id=2; DROP TABLE productos; --
- Objetivo: Si el mensaje de error indica que la tabla ha sido eliminada, tienes una vulnerabilidad seria.

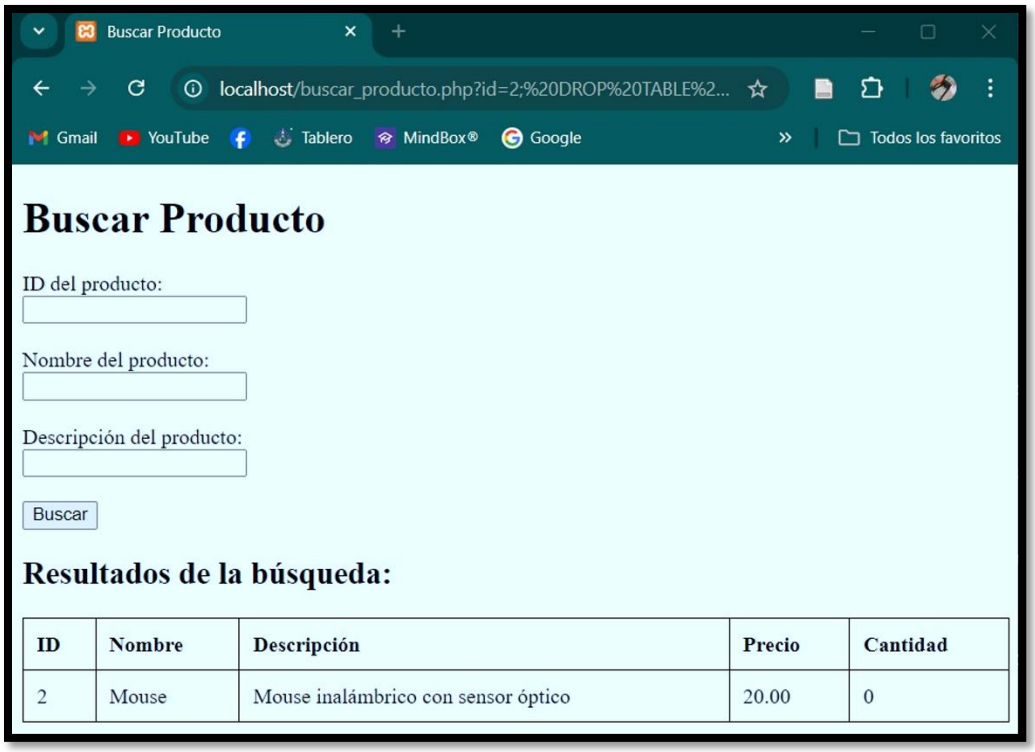

Ilustración 7: Inyección SQL de explotación de errores.

4. **Documentar los resultados obtenidos y las posibles acciones que se pueden realizar para prevenir este tipo de vulnerabilidades.**

Durante la creación y prueba de la aplicación web, se implementó un sistema de búsqueda que permite consultar registros por su ID, nombre o descripción. Las pruebas de funcionamiento demostraron que, al ingresar cualquier valor en estos campos, la aplicación responde con una tabla que contiene los registros coincidentes. Esta funcionalidad es esencial para la gestión de datos, ya que permite a los usuarios localizar información específica de manera eficiente.

Además, se realizaron pruebas de inyección de código SQL para evaluar la seguridad de la aplicación frente a este tipo de ataques. A través de ejemplos prácticos, se probaron diferentes formas de inyección:

- **Prueba básica de inyección SQL:** Si introduce una coma simple (') en el campo de búsqueda, lo que produce un error en la aplicación. Esto reveló

que el sistema no estaba manejando adecuadamente las entradas del usuario, dejándolo vulnerable a ataques.

- **Inyección SQL básica:** Se utilizó la inyección OR 1=1para intentar alterar la consulta. Esta técnica desarrolló todos los registros de la tabla, demostrando que la aplicación es vulnerable a inyecciones SQL. Si un atacante explota esta vulnerabilidad, podría obtener acceso a todos los datos almacenados, comprometiendo la seguridad del sistema.
- **Explotación de error a través de SQL:** Se intentó un ataque más agresivo mediante la inyección DROP TABLE productos. Aunque la aplicación devolvió un error, no se eliminó la tabla, lo que indica que la base de datos cuenta con ciertas protecciones, aunque aún hay vulnerabilidades presentes.

Acciones para prevenir la vulnerabilidad:

- Asegurarse de que todos los datos de entrada sean validados y sanitizados antes de procesarlos.
- En lugar de concatenar entradas en consultas SQL, utilizar consultas preparadas.
- Limitar los permisos y asegurar que los usuarios solo puedan acceder a los datos que les corresponden.
- Considerar el uso de firewalls de aplicaciones web (WAF) y otras herramientas de seguridad.

INVESTIGACIÓN.

ÍNDICE.

ÍNDICE DE ILUSTRACIONES

1. INYECCIÓN DE SQL

Ilustración 8: Inyección De Sql.

1.1 DEFINICIÓN

Una inyección de SQL, 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.

Como consecuencias de estos ataques y dependiendo de los privilegios que tenga el usuario de la base de datos bajo el que se ejecutan las consultas, se podría acceder no sólo a las tablas relacionadas con la aplicación, sino también a otras tablas pertenecientes a otras bases de datos alojadas en ese mismo servidor.

2. BLIND SQL INJECTION

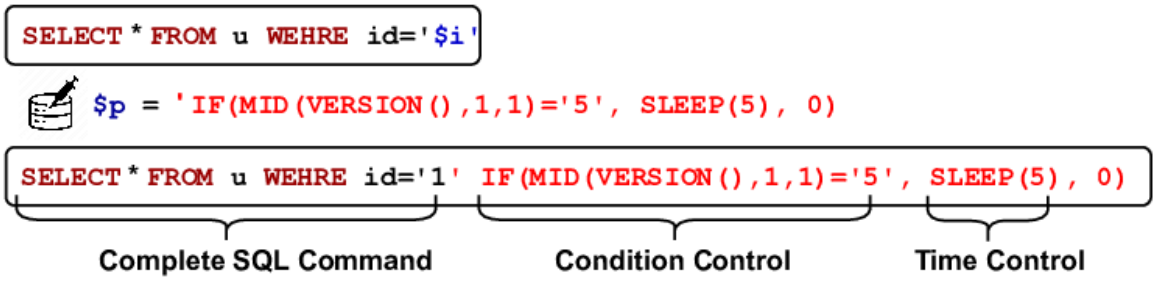

Ilustración 9: Inyección Ciega De Sql.

En este tipo de SQLi, los atacantes usan patrones de respuesta y comportamiento del servidor después de enviar cargas útiles de datos para obtener más información sobre su estructura. Los datos no se transfieren de la base de datos del sitio web al atacante, así que el atacante no ve la información sobre el ataque en banda (por eso se usa el término “SQLi ciega”). La SQLi inferencial se puede clasificar en dos subtipos:

2.1 SQLI BASADO EN EL TIEMPO:

Los atacantes envían una consulta de SQL a la base de datos, esto hace que la base de datos espere unos segundos antes de responder si la consulta es verdadera o falsa. Con este método, un atacante enumera cada letra del dato deseado utilizando la siguiente lógica:

- Si la primera letra del nombre de la primera base de datos es una "A", espere 10 segundos.
- Si la primera letra del nombre de la primera base de datos es una 'B', espere 10 segundos, etc.

Consulta inyectada:

Supongamos que la URL vulnerable es algo como:

http://vulnerable-site.com/product.php?id=1

El atacante puede inyectar la siguiente consulta para verificar si la primera letra del nombre de la base de datos es 'A':

1' AND IF(SUBSTRING((SELECT DATABASE()), 1, 1) = 'A', SLEEP(10), 0)--

Si la página tarda 10 segundos en responder, el atacante sabrá que la primera letra del nombre de la base de datos es 'A'. Si no hay retraso, el atacante puede probar con la siguiente letra, 'B':

```
1' AND IF(SUBSTRING((SELECT DATABASE()), 1, 1) = 'B', SLEEP(10), 0)--
```

El atacante realiza inyecciones para comprobar cada letra del alfabeto.

Por cada letra probada, si el servidor tarda 10 segundos en responder, el atacante sabe que ha encontrado la letra correcta.

Repitiendo este proceso, el atacante puede reconstruir el nombre completo de la base de datos, probando letra por letra.

2.2 SQLI BOOLEANA:

Los atacantes envían una consulta de SQL a la base de datos, así permiten que la aplicación responda mediante la generación de un resultado verdadero o falso.

Ejemplo:

```
SELECT * FROM productos WHERE id = 1 AND 1=1 --;
```

```
SELECT * FROM productos WHERE id = 1 AND 1=0 --;
```

El atacante puede modificar 1=1 o 1=0 para observar el comportamiento de la aplicación.

3. SQL INJECTION BASADA EN ERRORES

Las inyecciones de SQL basadas en errores aprovechan los mensajes de error de la base de datos para revelar información sensible. Los atacantes que están detrás de estos ataques envían intencionadamente consultas SQL malformadas, haciendo que la base de datos genere mensajes de error que contienen información valiosa. Analizando detenidamente estos mensajes, los ciberdelincuentes pueden conocer el funcionamiento interno del sistema e identificar sus (de los sistemas) posibles puntos débiles.

Los atacantes pueden usar los mensajes de error para descubrir información sobre la base de datos, como:

- **Estructura de tablas:** Los nombres de tablas y columnas pueden aparecer en los errores.
- **Versión del motor de la base de datos:** Algunos errores revelan información sobre la versión exacta del sistema de gestión de bases de datos (DBMS), lo que puede ser útil para explotar vulnerabilidades específicas de esa versión.
- **Tipos de datos:** Los errores pueden exponer los tipos de datos de las columnas, ayudando al atacante a formular consultas más precisas.

4. SQL INJECTION BASADA EN TIEMPO

Técnica de ataque en la que el atacante, al no poder obtener directamente los resultados de una consulta (como en los casos de blind SQL injection), usa funciones de retraso de tiempo para determinar si una consulta SQL es verdadera o falsa, basándose en el tiempo que tarda la aplicación en responder.

En este tipo de ataque, se inyectan comandos SQL que, si son verdaderos, hacen que el servidor retrase su respuesta durante un período de tiempo (usualmente usando funciones como SLEEP (), BENCHMARK (), etc.). El atacante entonces puede deducir información sobre la base de datos basándose en cuánto tarda la respuesta.

5. SQL INJECTION EN PROCEDIMIENTOS ALMACENADOS

SQL Injection en procedimientos almacenados se refiere a la vulnerabilidad que ocurre cuando los procedimientos almacenados en una base de datos no se implementan de manera segura, lo que permite a los atacantes inyectar comandos SQL maliciosos. Un procedimiento almacenado es un conjunto de consultas SQL predefinidas que pueden ejecutarse dentro de la base de datos. Si estos procedimientos no gestionan adecuadamente las entradas de los usuarios, pueden ser vulnerables a ataques de inyección SQL.

¿Qué es un procedimiento almacenado?

Un procedimiento almacenado es un bloque de código SQL que se almacena y ejecuta en la base de datos. Se utiliza comúnmente para realizar operaciones repetitivas, mejorar el rendimiento y la seguridad, y evitar duplicar consultas SQL en la aplicación.

6. SQL INJECTION EN ORM

El ORM es un modelo de programación que transforma las tablas de una base de datos en entidades para simplificar en gran medida la tarea del programador y acelerar el desarrollo de aplicaciones. Gracias al mapeo automático se puede cambiar de motor de base de datos fácilmente y cuando sea requerido.

Las acciones Insertar, Leer, Actualizar y Borrar a ejecutar sobre la base de datos se realizan de forma indirecta por medio del ORM.

Los ORM tienen el objetivo de liberarnos de la escritura o generación manual de código SQL necesario para realizar las consultas. Algunos ejemplos de estos son:

- Java
  - Hibernate
  - iBatis
  - Ebean
- .NET
  - nHibernate
  - Entity Framework
- PHP
  - Doctrine
  - Propel

6.1 ¿CÓMO OCURRE LA INYECCIÓN SQL EN ORM?

A pesar de que los ORM están diseñados para evitar inyecciones SQL, pueden seguir siendo vulnerables si el desarrollador introduce entradas de usuario de manera insegura o usa consultas SQL personalizadas (RAD SQL). Algunos errores comunes que pueden llevar a una inyección SQL en ORM son:

Consultas SQL crudas (RAD SQL): Muchos ORM permiten ejecutar consultas SQL manualmente. Si estas consultas no están parametrizadas o las entradas del usuario no son debidamente validadas, pueden estar expuestas a inyecciones SQL.

Filtrado dinámico basado en entrada del usuario: Si los desarrolladores permiten que los usuarios construyan dinámicamente las consultas de búsqueda o filtrado sin validación adecuada, también pueden exponer la aplicación a ataques.

Interpolación de cadenas: El uso de interpolación de cadenas o concatenar.

7. HERRAMIENTAS PARA DETECTAR Y PREVENIR SQL INJECTION

- **SQLMap:** una herramienta de pruebas de penetración de código abierto que se usa para detectar vulnerabilidades de inyección de SQL en aplicaciones web. La usan mucho los profesionales de la seguridad porque es muy personalizable y admite varios sistemas de gestión de bases de datos, como Microsoft SQL Server y Oracle.
- **jSQL Injection:** otra popular herramienta de código abierto diseñada para probar aplicaciones web. Esta herramienta ayuda a mejorar la ciberseguridad al permitir a los profesionales de TI y a los hackers éticos simular ataques, identificar vulnerabilidades y comprender el impacto potencial de las mismas.
- **Imperva:** una empresa llamada Imperva ofrece una herramienta llamada Imperva SQL Injection Protection que ayuda a las empresas a mantener sus

bases de datos a salvo de ataques de inyección de SQL. Proporciona supervisión en tiempo real y bloquea la actividad maliciosa, al tiempo que ofrece información detallada e informes sobre cualquier intento de ataque, lo que ayuda a las organizaciones a comprender y abordar rápidamente los riesgos de SQL.

- **AppSpider:** es una sólida herramienta de pruebas de seguridad de aplicaciones web que cuenta con detección de inyección de SQL. Gracias a su interfaz intuitiva, AppSpider permite a los usuarios gestionar y supervisar las vulnerabilidades de forma eficaz, al ofrecer informes detallados y recomendaciones de corrección.
- **Acunetix:** una herramienta más popular de inyección de SQL que ayuda a las organizaciones a auditar sus aplicaciones web e identificar vulnerabilidades de inyección de SQL. Sus completas funciones de escaneado (parcialmente basadas en el aprendizaje automático) y su interfaz fácil de usar la convierten en una solución vital para mantener una seguridad web sólida.

8. CONCLUSIÓN:

A través de esta práctica, aprendimos cómo las inyecciones SQL son un riesgo en el desarrollo de aplicaciones web. Estas vulnerabilidades surgen cuando las entradas del usuario no son correctamente validadas o filtradas, lo que permite a un atacante ejecutar comandos no autorizados en la base de datos. En este caso, al alterar la consulta SQL, un atacante puede extraer datos sensibles o incluso dañar la base de datos. Es importante implementar medidas de seguridad, como la parametrización de consultas y la validación rigurosa de las entradas del usuario, para prevenir ataques. Las inyecciones SQL son peligrosas porque pueden comprometer la confidencialidad, integridad y datos, afectando tanto a los usuarios como a la organización responsable del sistema, proteger una aplicación web de inyecciones SQL es esencial para garantizar la seguridad de los datos y evitar la explotación por parte de atacantes malintencionados.

9. REFERENCIAS BIBLIOGRAFICAS:

- *Ataques de inyección SQL, Qué son y cómo protegerse.* (s.f.). Hostalia: https://pressroom.hostalia.com/contents/ui/theme/images/inyeccion-sql-wp-hostalia.pdf
- Kaspersky. (s.f.). Retrieved 02 de 10 de 2024, from https://latam.kaspersky.com/resource-center/definitions/sql-injection
- nordpass. (s.f.). *nordpass*. https://nordpass.com/es/blog/what-is-sql-injection/
- *OWASP*. (s.f.). Retrieved 02 de 10 de 2024, from https://owasp.org/www-community/attacks/Blind_SQL_Injection