



## **Tecnológico Nacional de México Instituto Tecnológico de Tlaxiaco**

**Carrera: Ingeniería en Sistemas Computacionales**

**Materia: Seguridad Y Virtualización**

**Tema: Introducción a la seguridad de la información**

**Actividad: Reporte de Practica 3**

<b>Alumnos:</b>	Feria Ortiz Eduardo Tomas	<b>21620095</b>
	Reyes Peña Isaí	<b>21620053</b>
	Zárate Reyes Irving	<b>20620166</b>

**Grupo: 6US**

**Catedrático: Ing. Osorio Salinas Edward**

*Heroica Ciudad de Tlaxiaco.  
Miércoles, 18 de septiembre de 2024.*





## Índice

<b>1.- Parte - Práctica.....</b>	<b>3</b>
<b>Modificar el archivo php.ini .....</b>	<b>3</b>
<b>2. Cargar la Base de Datos desde un Archivo CSV .....</b>	<b>5</b>
<b>3. Creación de Usuarios con Permisos Específicos .....</b>	<b>6</b>
<b>4. Inyección de Datos en la Tabla users.....</b>	<b>9</b>
<b>5. Creación y Restauración de un Respaldo de la Base de Datos .....</b>	<b>11</b>
<b>Parte 2 - Investigación .....</b>	<b>13</b>
<b>Injection: .....</b>	<b>13</b>
<b>Bases de datos seguras:.....</b>	<b>16</b>
<b>Bibliografía:.....</b>	<b>21</b>





## 1.- Parte - Práctica.

### Modificar el archivo php.ini

Para comenzar con la practica creamos la base de datos con el Scrip que se encontraba en el repositorio de github. Posteriormente tuvimos que hacer algunos ajustes para que el archivo .csv de ese tamaño pudiera ser importado. Y para esto seguimos los siguientes pasos.

1. Primero abrimos el Panel de Control de XAMPP.
2. Hicimos clic en el botón de configuración (Config) junto a Apache y selecciona php.ini.
3. Buscamos y ajustamos los siguientes parámetros:
  - upload\_max\_filesize = 500M
  - post\_max\_size = 500M
  - max\_execution\_time = 300
  - max\_input\_time = 300
  - memory\_limit = 512M
4. Guardamos los cambios y reinicia Apache desde el Panel de Control de XAMPP.

Una vez reiniciados los servicios de XAMPP, ya es posible cargar archivos de hasta 512 MB.





```
php: Bloc de notas
Archivo  Editar  Ver

;;;;;;;;;;;;;;;;;;;;;;;;;

; Maximum execution time of each script, in seconds
; http://php.net/max-execution-time
; Note: This directive is hardcoded to 0 for the CLI SAPI
max_execution_time=600

; Maximum amount of time each script may spend parsing request data. It's a good
; idea to limit this time on productions servers in order to eliminate unexpectedly
; long running scripts.
; Note: This directive is hardcoded to -1 for the CLI SAPI
; Default Value: -1 (Unlimited)
; Development Value: 60 (60 seconds)
; Production Value: 60 (60 seconds)
; http://php.net/max-input-time
max_input_time=600

; Maximum input variable nesting level
; http://php.net/max-input-nesting-level
;max_input_nesting_level = 64

Ln 1, Col 1      100%      Windows (CRLF)      UTF-8

php: Bloc de notas
Archivo  Editar  Ver

upload_tmp_dir="C:\xampp\tmp"

; Maximum allowed size for uploaded files.
; http://php.net/upload-max-filesize
upload_max_filesize=512M

; Maximum number of files that can be uploaded via a single request
max_file_uploads=20

;;;;;;;;;;;;;;;;;;;;;;;;;
; Fopen wrappers ;
;;;;;;;;;;;;;;;;;;;;;;;;;

; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-fopen
allow_url_fopen=0n

; Whether to allow include/require to open URLs (like http:// or ftp://) as files
; http://php.net/allow-url-include
allow_url_include=Off
```

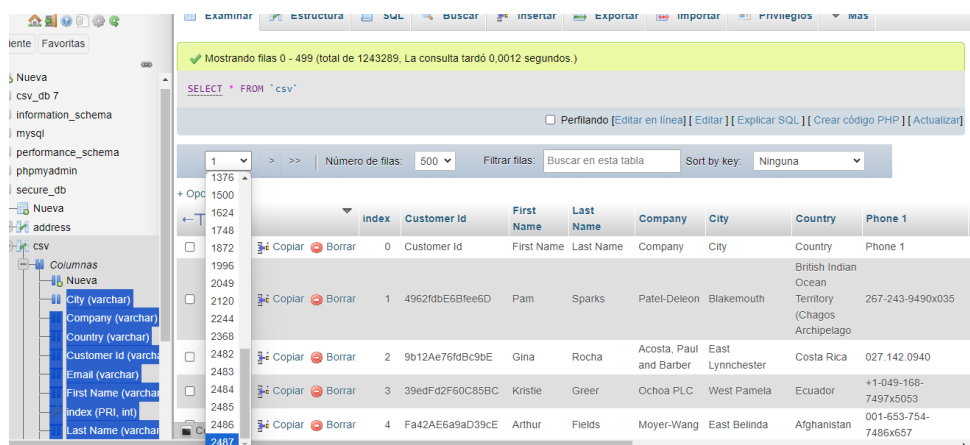


## 2. Cargar la Base de Datos desde un Archivo CSV

El siguiente paso consiste en subir datos a la base de datos desde un archivo CSV utilizando un script en Python. Para ello, definimos el procedimiento:

1. **Conexión a la base de datos:** Conectamos el script a la base de datos `secure_db` a través de las credenciales y demás parámetros de configuración.
2. **Lectura del archivo CSV:** El archivo `cm.csv`, que contiene los datos de clientes, es leído usando la MySQL Connection
3. **Inserción de datos:** Los datos del archivo CSV fueron insertados en la tabla correspondiente de la base de datos.
4. **Confirmación y cierre:** Una vez insertados todos los datos, se confirma la transacción y se cierra la conexión a la base de datos.

Como resultado los datos del CSV fueron importados correctamente a la base de datos.



Mostrando filas 0 - 499 (total de 1243289. La consulta tardó 0.0012 segundos.)

SELECT \* FROM `customers`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Número de filas: 500 Filtar filas: Buscar en esta tabla Sort by key: Ninguna

	index	Customer Id	First Name	Last Name	Company	City	Country	Phone 1
		0	Customer Id	First Name	Last Name	Company	City	Country
		1	49621dbE6Bf6e6D	Pam	Sparks	Patel-Deleon	Blakemouth	British Indian Ocean Territory (Chagos Archipelago)
		2	9b12Ae76f6Bc9bE	Gina	Rocha	Acosta, Paul and Barber	East Lynchester	Costa Rica
		3	39edFd2F60C85BC	Kristie	Greer	Ochoa PLC	West Pamela	Ecuador
		4	Fa42AE6a9aD39cE	Arthur	Fields	Moyer-Wang	East Belinda	Afghanistan



	Customer Id	First Name	Last Name	Company	City	Country	Phone 1
<input type="checkbox"/>	1243493	Gerald	Leon	Sons	Lake Katlin	Spain	(88) 7690-24b4
<input type="checkbox"/>	1243494	Noah	Carroll	Ochoa, Hebert and Grant	Yorkport	Russian Federation	001-566-226-6518x602
<input type="checkbox"/>	1243495	Javier	Choi	Butler Ltd	East Parker	Norway	001-051-282-6285
<input type="checkbox"/>	1243496	Kathleen	McLaughlin	Wilkerson, Townsend and Finley	Billymouth	France	001-702-444-6822x066
<input type="checkbox"/>	1243497	Candice	Hinton	Johns, Parks and Gonzalez	North Brandy	Russian Federation	001-857-205-2034x627
<input type="checkbox"/>	1243498	Warren	Wheeler	Daniels-Lowe	Jeanshire	Guadeloupe	085-744-9552
<input type="checkbox"/>	1243499	Marcus	Callahan	Ali-Duffy	North Pedroshire	Madagascar	001-229-702-7494x256

### 3. Creación de Usuarios con Permisos Específicos

Se crearon tres usuarios en la base de datos secure\_db, cada uno con diferentes niveles de permisos:

- **Usuario 1:** Tiene permisos de solo lectura sobre la tabla customers.
- **Usuario 2:** Tiene permisos de lectura y escritura sobre la tabla address, lo que permite leer y modificar datos en esta tabla.
- **Usuario 3:** Tiene permisos de lectura, escritura y eliminación sobre la tabla users, lo que le permite manipular completamente esta tabla.



Servidor: 127.0.0.1

Bases de datos

SQL

Estado actual

Cuentas de usuarios

Exportar

Importar

Configuración

Más

Base de datos

Editar los privilegios: Cuenta de usuario 'user1'@'localhost' - Bases de datos secure\_db

Privilegios específicos para la base de datos ☐ Seleccionar todo

Nota: Los nombres de los privilegios de MySQL están expresados en inglés.

☒ Datos

☐ Estructura

☐ Administración

☒ SELECT

☒ INSERT

☒ UPDATE

☒ DELETE

☐ CREATE

☐ ALTER

☐ INDEX

☐ DROP

☐ CREATE TEMPORARY TABLES

☐ GRANT

☐ LOCK TABLES

☐ REFERENCES



Servidor: 127.0.0.1

Bases de datos

SQL

Estado actual

Cuentas de usuarios

Exportar

Importar

Configuración

Más

Ha agregado un nuevo usuario.

```
CREATE USER 'user2'@'admin' IDENTIFIED VIA mysql_native_password USING '****';GRANT SELECT, INSERT, UPDATE, DELETE, FILE ON *.* TO 'user2'@'admin' REQUIRE NONE WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;
```

[Editar en línea] [Editar] [Crear código PHP]

Global

Base de datos

Cambio de contraseña

Información de la cuenta

Editar los privilegios: Cuenta de usuario 'user2'@'admin'

Privilegios específicos para la base de datos

Base de datos

Privilegios

Conceder

Privilegios específicos para la tabla

Acción

Ninguna

csv\_db 7

mysql

phpmyadmin

secure\_db

Añadir privilegios a la o las base de datos siguientes:

Consola

Favoritos

Opciones

Historial

Limpiar

Servidor: 127.0.0.1

Bases de datos

SQL

Estado actual

Cuentas de usuarios

Exportar

Importar

Configuración

Más

Ha agregado un nuevo usuario.

```
CREATE USER 'user3'@'admin5' IDENTIFIED VIA mysql_native_password USING '****';GRANT SELECT, INSERT, UPDATE, DELETE, FILE ON *.* TO 'user3'@'admin5' REQUIRE NONE WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;
```

[Editar en línea] [Editar] [Crear código PHP]

Global

Base de datos

Cambio de contraseña

Información de la cuenta

Editar los privilegios: Cuenta de usuario 'user3'@'admin5'

Privilegios específicos para la base de datos

Base de datos

Privilegios

Conceder

Privilegios específicos para la tabla

Acción

Ninguna

mysql

phpmyadmin

secure\_db

Consola

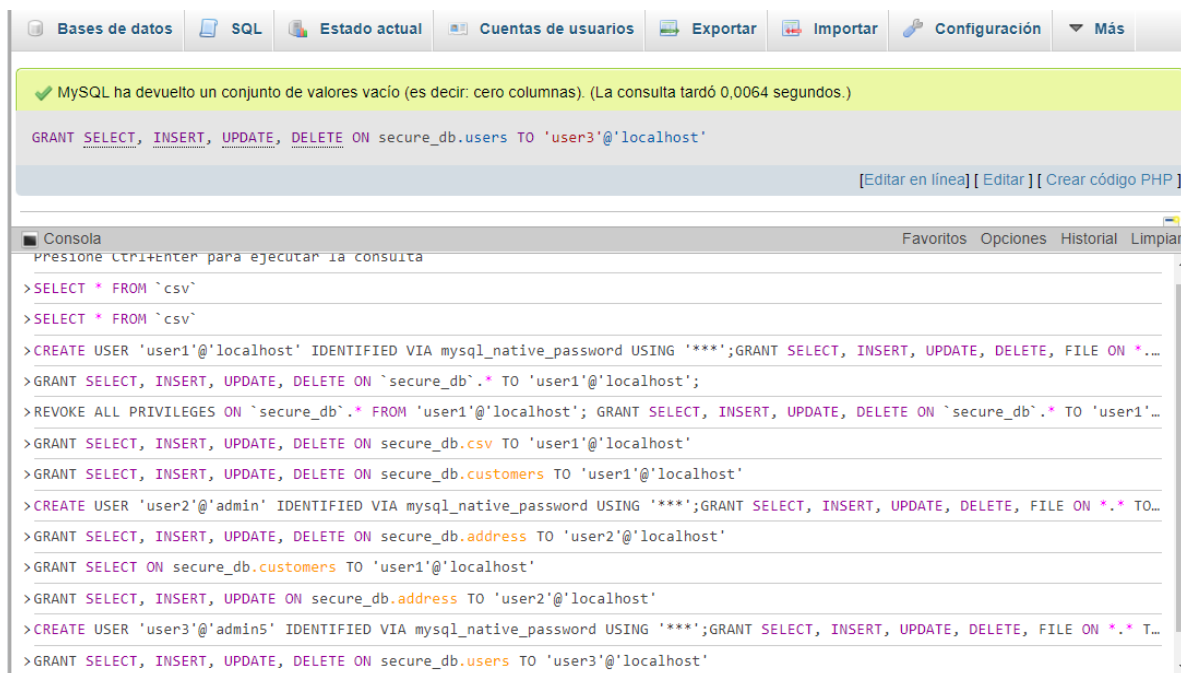
Favoritos

Opciones

Historial

Limpiar





```
MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0064 segundos.)

GRANT SELECT, INSERT, UPDATE, DELETE ON secure_db.users TO 'user3'@'localhost';

[Editar en línea] [Editar] [Crear código PHP]
```

Consola

Presione Ctrl+Enter para ejecutar la consulta

```
>SELECT * FROM `csv`
>SELECT * FROM `csv`
>CREATE USER 'user1'@'localhost' IDENTIFIED VIA mysql_native_password USING '***';GRANT SELECT, INSERT, UPDATE, DELETE, FILE ON *.* TO...
>GRANT SELECT, INSERT, UPDATE, DELETE ON `secure_db`. * TO 'user1'@'localhost';
>REVOKE ALL PRIVILEGES ON `secure_db`. * FROM 'user1'@'localhost'; GRANT SELECT, INSERT, UPDATE, DELETE ON `secure_db`. * TO 'user1'...
>GRANT SELECT, INSERT, UPDATE, DELETE ON secure_db.csv TO 'user1'@'localhost'
>GRANT SELECT, INSERT, UPDATE, DELETE ON secure_db.customers TO 'user1'@'localhost'
>CREATE USER 'user2'@'admin' IDENTIFIED VIA mysql_native_password USING '***';GRANT SELECT, INSERT, UPDATE, DELETE, FILE ON *.* TO...
>GRANT SELECT, INSERT, UPDATE, DELETE ON secure_db.address TO 'user2'@'localhost'
>GRANT SELECT ON secure_db.customers TO 'user1'@'localhost'
>GRANT SELECT, INSERT, UPDATE ON secure_db.address TO 'user2'@'localhost'
>CREATE USER 'user3'@'admins' IDENTIFIED VIA mysql_native_password USING '***';GRANT SELECT, INSERT, UPDATE, DELETE, FILE ON *.* TO...
>GRANT SELECT, INSERT, UPDATE, DELETE ON secure_db.users TO 'user3'@'localhost'
```

#### 4. Inyección de Datos en la Tabla users

Para demostrar una inyección SQL, se ejecutó un script que realiza una consulta en la tabla users, solicitando el nombre de usuario como entrada. Este ejemplo muestra cómo una inyección SQL puede afectar la seguridad de la base de datos si no se gestionan adecuadamente las entradas de usuario.

Proceso:

1. **Conexión a la base de datos:** Se establece una conexión con la base de datos secure\_db.



2. **Consulta SQL:** Se ejecuta una consulta para obtener los datos del usuario según el nombre introducido.
3. **Resultados:** El script imprime en consola los datos obtenidos de la tabla users.
4. **Manejo de errores:** Se gestionan posibles errores de conexión o de acceso a la base de datos.

```
> Users > sn > Desktop > import mysql.py > ...
1 import mysql.connector
2 from mysql.connector import errorcode
3
4 try:
5     conn = mysql.connector.connect(
6         user='root',
7         password='',
8         host='localhost',
9         database='secure_db'
10    )
11    cursor = conn.cursor()
12
13    # Consulta segura usando parámetros
14    query = "SELECT * FROM users WHERE username = %s"
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console

```
C:\Users\sn\Desktop> ss> & 'c:\Users\sn\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\sn\.vscode\extensions\m
-python.debugpy-2024.10.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher' '50865' '--' 'C:\Users\sn\Desktop\impo
t mysql.py'
```



## 5. Creación y Restauración de un Respaldo de la Base de Datos

Para garantizar la integridad y disponibilidad de la base de datos, se creó un respaldo (backup) y se restauró en un servidor diferente:

1. **Respaldo de la base de datos:** Se generó un archivo de respaldo `secure_db_backup.sql` utilizando el comando `mysqldump`.
2. **Restauración en otro servidor:** El archivo de respaldo fue transferido al nuevo servidor mediante `scp`, y posteriormente restaurado en la nueva base de datos.

Este proceso garantiza que los datos de la base de datos estén disponibles y seguros en caso de fallos en el servidor original.





```
XAMPP for Windows

Setting environment for using XAMPP for Windows.
n@ISAI c:\xampp
mysql> mysql -u [usuario] -p [nombre_de_base_de_datos] > [ruta_del_archivo]/secure_db_backup.sql
El sistema no puede encontrar la ruta especificada.

n@ISAI c:\xampp
mysql> mysql -u [usuario] -p [nombre_de_base_de_datos] > /secure_db_backup.sql
Acceso denegado.

n@ISAI c:\xampp
mysql> mysql -u [usuario] -p [nombre_de_base_de_datos] > /secure_db_backup.sql
Acceso denegado.

n@ISAI c:\xampp
mysql> mysql -u root -p secure_db > C:\Users\sn\Desktop\csv\secure_db_backup.sql
Enter password:
n@ISAI c:\xampp
```

C:\Users\sn\Desktop\csv

Nombre	Fecha de modificación	Tipo	Tamaño
cm	30/05/2022 10:44 a. m.	Archivo de valores...	341,228 KB
customers-2000000	17/09/2024 08:15 p. m.	Archivo WinRAR	200,169 KB
secure_db_backup	18/09/2024 04:50 p. m.	Archivo SQL	255,302 KB

## Parte 2 - Investigación

Investiga y describe los conceptos de SQL Injection y cómo se pueden prevenir.

### Injection:

SQL Injection (Inyección de SQL) es una vulnerabilidad de seguridad que permite a un atacante alterar las consultas que una aplicación realiza a su base de datos. Esto ocurre cuando los datos proporcionados por el usuario no se gestionan correctamente y se incluyen directamente en una consulta SQL. De esta forma, el atacante puede modificar la consulta original, logrando acceder, modificar o eliminar información en la base de datos, comprometiendo su integridad. Este tipo de ataque es posible cuando las entradas de usuario no son validadas adecuadamente. Un ejemplo típico es cuando se permite que un atacante inyecte código SQL en campos de entrada, como formularios de inicio de sesión, lo que puede provocar la ejecución de consultas inesperadas y no deseadas.

Existen varios tipos de inyección SQL, siendo el más común el ataque de inyección clásica o de texto, donde el atacante inserta código malicioso directamente en los campos de entrada. Otro tipo es la inyección basada en el tiempo, donde se usan consultas SQL que introducen retrasos, lo que permite al atacante medir el tiempo de respuesta del servidor y deducir información valiosa. Por último, la inyección ciega (Blind SQL Injection) se da cuando no se reciben respuestas visibles de la base de datos, pero el atacante puede inferir el comportamiento a





partir de pequeñas diferencias en los tiempos de respuesta o en el comportamiento de la aplicación.

### **Tipos de SQL Injection**

- Inyección Clásica o de Texto: Ocurre cuando un atacante inyecta código SQL directamente en los campos de entrada, como formularios o URL.
- Inyección Basada en el Tiempo: Se utilizan consultas SQL que introducen retrasos deliberados, midiendo cuánto tarda en responder la base de datos para inferir información.
- Inyección Ciega (Blind SQL Injection): No se recibe una respuesta directa de la base de datos, pero se obtienen datos explotando diferencias en los tiempos de respuesta o comportamientos sutiles.

Para prevenir este tipo de ataques, es fundamental adoptar varias prácticas de seguridad. El uso de consultas preparadas (Prepared Statements) es una de las mejores maneras de proteger las bases de datos. Estas consultas separan los datos de los comandos SQL, lo que impide que los valores ingresados por el usuario se traten como parte del código SQL. Otra medida efectiva es la validación y saneamiento de las entradas, es decir, verificar que los datos ingresados sean del tipo y formato esperado. Esto reduce las posibilidades de que se inyecte código malicioso en las consultas.





Otra estrategia recomendada es implementar el principio de mínimos privilegios, asegurando que los usuarios solo tengan acceso a los recursos necesarios. De esta manera, se limita el daño que un atacante puede causar en caso de una inyección SQL exitosa. También se recomienda evitar mostrar mensajes de error detallados, ya que estos podrían proporcionar al atacante información clave sobre la estructura de la base de datos. Finalmente, el uso de un firewall para aplicaciones web (WAF) puede ayudar a detectar y bloquear intentos de inyección SQL, filtrando el tráfico malicioso antes de que llegue a la base de datos.

### **Consecuencias de una SQL Injection**

- **Robo de Información:** Un atacante puede obtener datos confidenciales, como credenciales de usuario, direcciones o información financiera.
- **Modificación de Datos:** Puede alterar o eliminar datos críticos, corrompiendo la integridad de la base de datos.
- **Acceso Administrativo:** En algunos casos, puede otorgar privilegios elevados a usuarios no autorizados, permitiendo el control completo de la base de datos.
- **Negación de Servicio (DoS):** El atacante puede bloquear el acceso legítimo a la base de datos, afectando la disponibilidad del sistema.





Investiga y describe los conceptos de bases de datos seguras y cómo se pueden implementar.

### **Bases de datos seguras:**

Una base de datos segura es aquella que ha sido diseñada, configurada y administrada con medidas de seguridad que protegen los datos almacenados contra accesos no autorizados, manipulaciones indebidas, pérdidas o ataques. La seguridad de una base de datos es crucial para preservar la confidencialidad, integridad y disponibilidad de la información. A continuación, se describen los conceptos clave para garantizar la seguridad en bases de datos y cómo implementarlos eficazmente.

### **Principios Clave de una Base de Datos Segura**

- **Confidencialidad:** Asegura que los datos solo sean accesibles para aquellos que estén autorizados a verlos. Esto incluye la protección contra accesos no autorizados a los datos sensibles, como información personal o financiera.
- **Integridad:** Garantiza que los datos no sean alterados de manera incorrecta o no intencionada. Esto implica que los datos deben ser precisos y mantenerse intactos durante su almacenamiento, transferencia y manipulación.
- **Disponibilidad:** Asegura que los usuarios autorizados puedan acceder a la base de datos y a los datos cuando sea necesario, evitando caídas o interrupciones que impidan su uso.







## Medidas de Seguridad en Bases de Datos

- **Control de Acceso Basado en Roles (RBAC):** Una de las estrategias más comunes para proteger una base de datos es asignar permisos específicos a los usuarios según su rol dentro de la organización. Mediante RBAC, los usuarios solo pueden acceder a las partes de la base de datos que son necesarias para realizar su trabajo, reduciendo así el riesgo de acceso no autorizado a datos sensibles.
- **Cifrado de Datos:** Tanto los datos almacenados (en reposo) como los datos en tránsito deben estar cifrados. El cifrado asegura que, incluso si los datos son interceptados o robados, no podrán ser leídos ni utilizados sin las claves de descifrado adecuadas.
  - **Cifrado en reposo:** Protege los datos almacenados en discos duros, bases de datos o cualquier otra forma de almacenamiento persistente.
  - **Cifrado en tránsito:** Protege los datos mientras se están transfiriendo entre sistemas, por ejemplo, en comunicaciones cliente-servidor o entre diferentes servicios de una aplicación.
- **Autenticación y Autorización:** La autenticación asegura que solo los usuarios autorizados puedan acceder a la base de datos. Esto puede implementarse con contraseñas seguras, autenticación multifactor (MFA) o certificados digitales. La autorización, por otro lado, determina qué acciones puede realizar un usuario autenticado dentro de la base de datos,





asegurando que cada usuario solo pueda acceder a los recursos que necesita.

- Auditoría y Monitoreo: Es esencial implementar registros de auditoría para mantener un seguimiento detallado de todas las actividades que se realizan en la base de datos. Esto incluye el acceso a los datos, cambios en la estructura de la base de datos y cualquier intento de acceso no autorizado. El monitoreo constante permite identificar comportamientos sospechosos o inusuales que puedan indicar intentos de ataque o uso indebido de la información.
- Parches y Actualizaciones de Seguridad: Mantener el software de la base de datos actualizado es crucial para evitar vulnerabilidades conocidas que puedan ser explotadas por atacantes. Los fabricantes de software liberan parches de seguridad regularmente, y es importante aplicarlos de manera oportuna.
- Copia de Seguridad (Backup): Las copias de seguridad periódicas de la base de datos garantizan que los datos puedan recuperarse en caso de pérdida, ataque o corrupción. Las copias deben almacenarse en un lugar seguro y deben estar protegidas mediante cifrado. Además, es fundamental realizar pruebas periódicas para asegurarse de que las copias de seguridad puedan restaurarse correctamente.
- Prevención de Inyección SQL: Una de las vulnerabilidades más comunes en las bases de datos es la inyección SQL. Para prevenirla, es esencial utilizar





consultas preparadas, validación de entradas y limpieza de datos en las aplicaciones que interactúan con la base de datos. Las consultas preparadas aseguran que los datos ingresados por el usuario no se interpreten como código SQL, evitando así que un atacante modifique la consulta original.

- **Segregación de Redes y Bases de Datos:** Mantener las bases de datos en redes separadas o segmentadas reduce la superficie de ataque. Las bases de datos críticas no deben estar accesibles directamente desde Internet, sino a través de servidores de aplicaciones o proxies, con capas de seguridad adicionales, como firewalls y sistemas de detección de intrusos (IDS).

### Implementación de Seguridad en Bases de Datos

- **Establecer Políticas de Seguridad:** Las organizaciones deben desarrollar políticas de seguridad claras que establezcan quién tiene acceso a qué datos y bajo qué circunstancias. Estas políticas deben incluir reglas sobre el manejo de datos sensibles, gestión de contraseñas y respuesta a incidentes de seguridad.
- **Testeo de Seguridad Regular:** Se deben realizar pruebas de penetración y análisis de vulnerabilidades en las bases de datos de forma periódica para identificar y corregir posibles debilidades antes de que puedan ser explotadas.





- Control de Acceso Físico: Aunque gran parte de la seguridad se enfoca en el entorno digital, es importante asegurarse de que el acceso físico a los servidores y sistemas que almacenan las bases de datos esté controlado. Solo el personal autorizado debería poder acceder a las instalaciones donde están alojadas las bases de datos.





## Bibliografía:

- Shar, L. K., & Tan, H. B. K. (2012). Defeating SQL injection. *Computer*, 46(3), 69-77.
- Sadeghian, A., Zamani, M., & Abdullah, S. M. (2013, September). A taxonomy of SQL injection attacks. In 2013 International Conference on Informatics and Creative Multimedia (pp. 269-273). IEEE.
- Clarke-Salt, J. (2009). SQL injection attacks and defense. Elsevier.
- Avilés, G. G. (2015). *Seguridad en bases de datos y aplicaciones web*. IT Campus Academy.
- Martínez, D. E. R., & Mora, H. M. (2014). Criptografía en bases de datos en cloud computing. *Aibi revista de investigación, administración e ingeniería*, 2(1), 45-54.
- Ibáñez, L. H. (2016). *Administración de sistemas gestores de bases de datos*. Ra-ma Editorial.

