TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TLAXIACO

INTEGRANTES:

JULISSA MIGDALIA JOSE CRUZ

LUZ MARIA MENDOZA CORTES

ANGELES GONZÁLEZ MARTNEZ

ELISAHAD MAYTE LEON DE JESUS

DOCENTE:

ING.EDWARD OSORIO SALINAS

MATERIA:

SEGURIDAD Y VIRTUALIZACION

PRACTICA:

3(BASES DE DATOS SEGURAS)

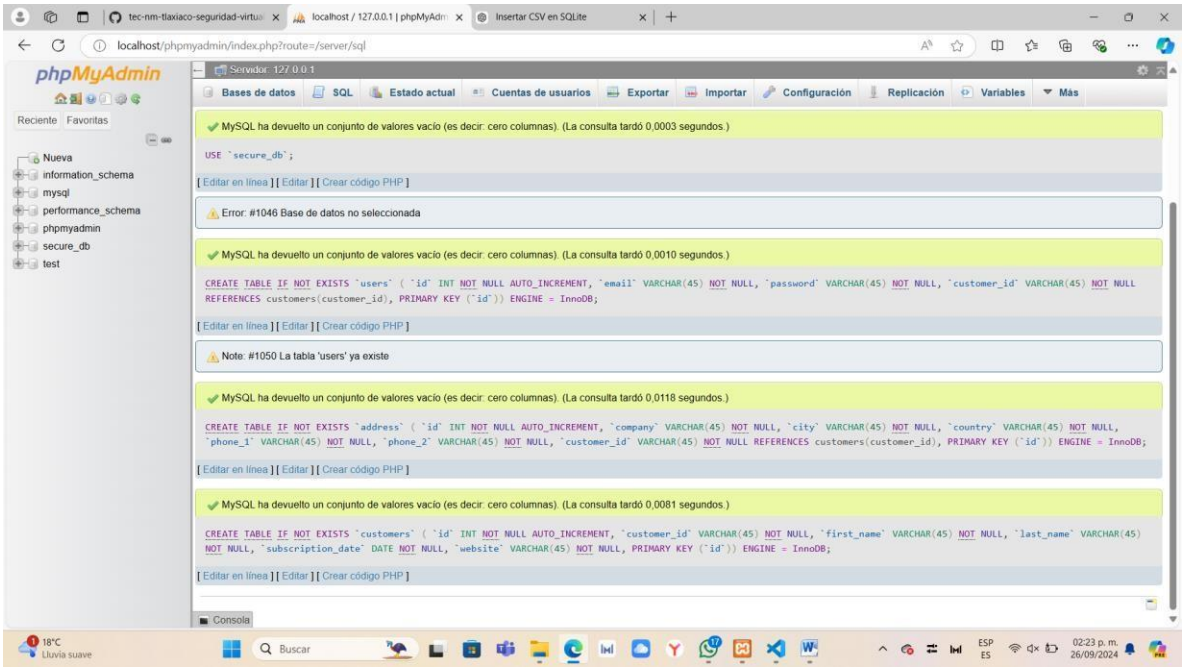
GRUPO:7US **SEMESTRE:** SEPTIMO

FECHA: 15 DE SEPTIEMBRE DEL 2024

INTRODUCCION:

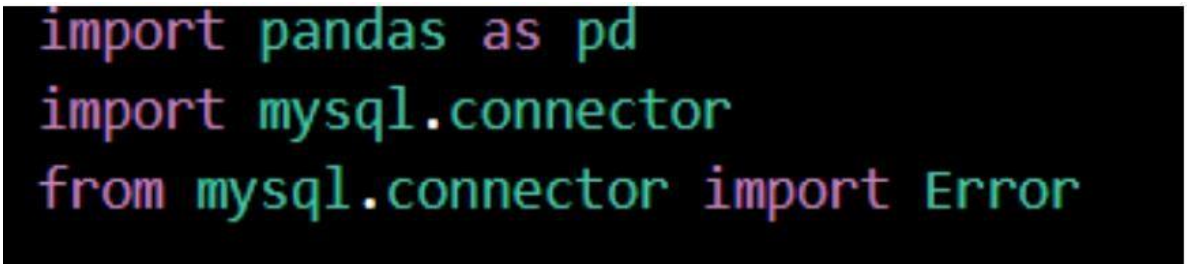
Las bases de datos son sistemas críticos para el almacenamiento, gestión y recuperación de grandes cantidades de información. En el entorno actual, donde los datos se han convertido en un activo valioso, la seguridad en las bases de datos es fundamental para proteger la confidencialidad, integridad y disponibilidad de la información almacenada. La práctica de **Bases de Datos Seguras** se enfoca en implementar medidas de seguridad que garanticen que los datos sean accesibles solo para usuarios autorizados, manteniéndolos protegidos de accesos no autorizados, modificaciones maliciosas o fugas de información. Entre las técnicas y herramientas que se abordan en esta práctica se encuentran el control de acceso basado en roles, la autenticación y autorización de usuarios, el cifrado de datos tanto en tránsito como en reposo, y el monitoreo de actividades dentro de la base de datos para detectar posibles amenazas o comportamientos sospechosos. Además, se estudian las vulnerabilidades más comunes que pueden ser explotadas por atacantes, como la inyección SQL y las malas configuraciones de seguridad. El objetivo principal es proporcionar una comprensión sólida de las mejores prácticas para asegurar una base de datos, asegurando que la información almacenada esté protegida contra riesgos internos y externos, respetando normativas y estándares de seguridad.

Principalmente se crea una base de datos en MySQL en XAMMP donde se insertaron 3 tablas con los siguientes campos, como se puede mostrar a continuación en la siguiente imagen:



2. Crea un script en Python que permita insertar los datos del archivo `customers-2000000.csv`:

Después nos dirigimos a lo que es VisualCode donde se realizo prácticamente lo que es un Script, donde importamos las librerías correspondientes ,para manipular datos, para importar el modulo y para importar la clase del modulo, como se puede observar a continuación en la siguiente imagen:



Después nos vamos a configurar la conexión de la base de datos, donde posteriormente se crea una db_config que contiene parámetros, los cuales ayudaran a establecer dicha conexión con la base de datos en el MySQL, como se puede observar a continuación en la siguiente imagen:

```
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '',  # Asegú
    'database': 'secure_db'
}
```

Después definimos lo que viene siendo la función para conectar la base de datos, donde nos mostrara si hay algún error al momento de realizar la conexión, como se puede observar a continuación en la siguiente imagen:

```
    try:
        connection = mysql.connector.connect(**db_config)
        if connection.is_connected():
            print("Conectado a la base de datos")
            return connection
    except Error as e:
        print(f"Error al conectar a la base de datos: {e}")
        return None
```

Después definimos una función, donde nos va a permitir insertar datos en el archivo CSV, como se puede mostrar a continuación en la siguiente imagen:

```
Tabnine: Edit | Test | Explain | Document | Ask
def insert_data_from_csv(csv_file_path):
    connection = connect_to_database()
    if connection is None:
        return
```

Después se realizan las consultas y mostramos líneas las cuakes per itiran insertar los datos de la tabla, como se puede observar a co tinuacion en la siguiente imagen:

```
insert_customers_query = """
INSERT INTO customers (customer_id, first_name, last_name, subscriptio
VALUES (%s, %s, %s, %s, %s)
```

Despues se utiliza lo que es INSERT_ADDRESS_QUERY para definir una consulta, como se puede mostrar a continuación en la siguiente imagen:

```
insert_address_query = """
INSERT INTO address (customer_id, company, city, country, phone_1, pho
VALUES (%s, %s, %s, %s, %s, %s)
"""
```

Después definimos lo que es una consulta SQL para insertar los datos en la TABLA USER, donde posteriormente se establece una contraseña, como se puede observar a continuación en la siguiente imagen:

```
insert_users_query = """
INSERT INTO users (customer_id, email, password)
VALUES (%s, %s, 'default_password')  # Puedes cambiar 'default_passwor
```

Posteriormente creamos lo que es una iteración de filas para el dataframe, donde nos va a permitir a realizar lo que es inserciones en el código , como se puede apreciar a continuación en la siguiente imagen:

```
for index, row in data.iterrows():
    try:
        # Insertar en la tabla 'customers'
        customer_tuple = (
            row['customer_id'],
            row['first_name'],
            row['last_name'],
            row['subscription_date'],
            row['website']
        )
        cursor.execute(insert_customers_query, customer_tuple)
```

Posteriormente se crea un **"address_tuple**", donde nos va a permitir realizar una consulta y por lo tanto insertar los datos en la tabla, como se puede mostrar a continuación en la siguiente imagen:

```
    # Insertar en la tabla 'address'
    address_tuple = (
        row['customer_id'],
        row['Company'],
        row['City'],
        row['Country'],
        row['Phone 1'],
        row['Phone 2']
    )
    cursor.execute(insert_address_query, address_tuple)
```

Después se crea otra consulta , la cual nos permitirá ejecutar la consulta SQL para insertar datos en la tabla, como se muestra a continuación;

```
    # Insertar en la tabla 'users'
    users_tuple = (
        row['customer_id'],
        row['Email']
    )
    cursor.execute(insert_users_query, users_tuple)
```

Después se agrega lo que es el manejo de los errores que habrá durante cuando inserten los datos, como se puede observara a continuación en la siguiente imagen:

```
except KeyError as e:
    print(f"Error de clave en la fila {index}: {e}")
except Error as e:
    print(f"Error al insertar los datos en la fila {index}: {e}")
```

Despues agregamos lo que es los cambios y cerra la conexión al momento de cerrar la base de datos, como se pude observar a continuación en la siguiente imagen:

```
connection.commit()
cursor.close()
connection.close()
print("Datos insertados exitosamente")
```

Y finalmente se define la ruta del archivo de CSV para iniciar el proceso en la base de datos, como se puede apreciar a continuación en la siguiente imagen:

```
csv_file_path = 'C:/xampp/mysql/data/secure_db/customers-2000000.csv'
insert_data_from_csv(csv_file_path)
```

Después se muestran los resultados donde se realiza la prueba donde se ve que está funcionando correctamente y posteriormente se ven los registros que se

están realizando dentro de la base de datos y posteriormente esta funcionando correctamente su conexión como se puede observar a continuación en las siguientes imágenes:


```
PS C:\xampp\mysql\data\secure_db>
/secure_db/conexion.py
Conectado a la base de datos
Columnas en el archivo CSV: Index(['Index', 'customer_id', 'first_name', 'last_name', 'Company', 'City',
       'Country', 'Phone 1', 'Phone 2', 'Email', 'subscription_date',
       'website'],
      dtype='object')
Datos insertados exitosamente
```

3. Crea tres usuarios en MySQL con los siguientes permisos:

Usuario 1: Permisos de lectura en la tabla `customers`

Usuario 2: Permisos de lectura y escritura en la tabla `address`

Usuario 3: Permisos de lectura, escritura y eliminación en la tabla `users`

Como paso número 1 realizamos lo que es que se creó el primer Usuario de lectura en la tabla , donde se especificó el nombre donde solo debe conectarse con el nombre de usuario 1, como se puede observar a continuación en la siguiente imagen:

```
-- Crear Usuario 1 con permisos de lectura en la tabla `customers`
CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';
GRANT SELECT ON secure_db.customers TO 'user1'@'localhost';
```

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0005 segundos.)

```
CREATE USER IF NOT EXISTS 'user1'@'localhost' IDENTIFIED BY 'password1';
```

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

Note: #1973 Can't create user 'user1'@'localhost'; it already exists

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0016 segundos.)

```
GRANT SELECT ON secure_db.customers TO 'user1'@'localhost';
```

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0016 segundos.)

```
CREATE USER IF NOT EXISTS 'user2'@'localhost' IDENTIFIED BY 'password2';
```

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

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0016 segundos.)

```
GRANT SELECT, INSERT, UPDATE ON secure_db.address TO 'user2'@'localhost';
```

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

Despues se creo lo que es el segundo usurio , donde se le asigno el campo que solicita la lectura en la tabla, donde se especifica el nombre de usuario, donde solo podrá conectarse desde la maquina local, donde el usuario pueda autentificarse como se puede mostrar a continuación en la siguiente imagen:

```
CREATE USER IF NOT EXISTS 'user2'@'localhost' IDENTIFIED BY 'password2';
GRANT SELECT, INSERT, UPDATE ON secure_db.address TO 'user2'@'localhost';
```

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0016 segundos.)

```
CREATE USER IF NOT EXISTS 'user2'@'localhost' IDENTIFIED BY 'password2';
```

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

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0016 segundos.)

```
GRANT SELECT, INSERT, UPDATE ON secure_db.address TO 'user2'@'localhost';
```

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

Y posteriormente se cera lo que es el 3 **“USUARIO”,** donde se le asigna otro campo que solicita de lectura en la tabla, donde se especifica el nombre del usuario donde se podrá conectarse desde una maquina local lo cual hará que el usuario pueda autentificarse de tal forma, como se puede observar a continuación en la siguiente imagen:

```
CREATE USER IF NOT EXISTS 'user3'@'localhost' IDENTIFIED BY 'password3';
GRANT SELECT, INSERT, UPDATE, DELETE ON secure_db.users TO 'user3'@'localhost';
FULL PRIVILEGES;
```

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0007 segundos.)

```
CREATE USER IF NOT EXISTS 'user3'@'localhost' IDENTIFIED BY 'password3';
```

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

⚠ Note: #1973 Can't create user 'user3'@'localhost'; it already exists

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0031 segundos.)

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

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

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0046 segundos.)

```
FLUSH PRIVILEGES;
```

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

4. Crea un script en Python que permita realizar una inyección de SQL en la tabla `users` y que muestre los datos de la tabla `users` en la consola.

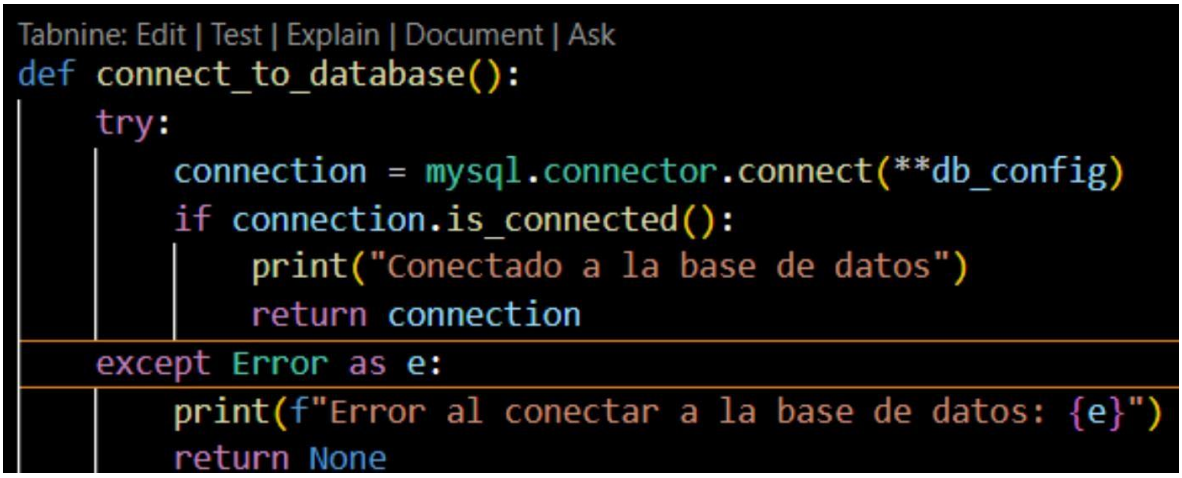
Principalmente importamos las librerías que posteriormente se van a utilizar, como se puede mostrar a continuación en la siguiente imagen:

```
import mysql.connector
from mysql.connector import Error
```

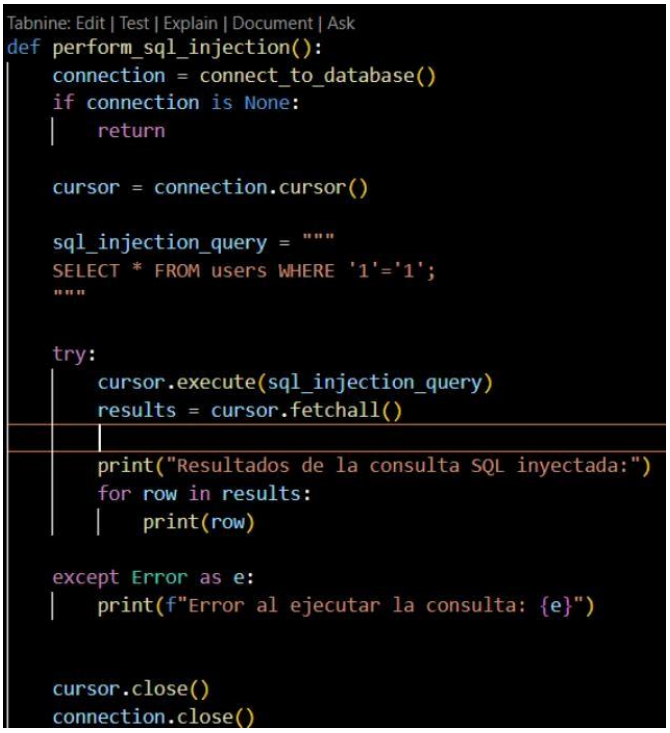
Después se configura la conexión a la base de datos como se puede observar a continuación en la siguiente imagen:

```
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'secure_db'
}
```

Posteriormente se crea una función la cual será para conectar nuestra base de datos, donde se intentara establecer una conexión mediante con sus respectivos parámetros para realizar la conexión y posteriormente nos mostrara un mensaje donde mostrara que la conexión fue exitosa o si la conexión tuvo algún fallo, como se puede ver a continuación en la siguiente imagen:



Después se realiza lo que será que se creo la función de injyection SQL , donde esta función manejara la conexión a la base de datos y posteriormente se realiza la consulta SQL que simula la inyección SQL como se puede observar a continuación en la siguiente imagen:



Después posteriormente se crea un bloque principal donde por lo cual se llama la inyección SQL, como se puede observar a continuación en la siguiente imagen:

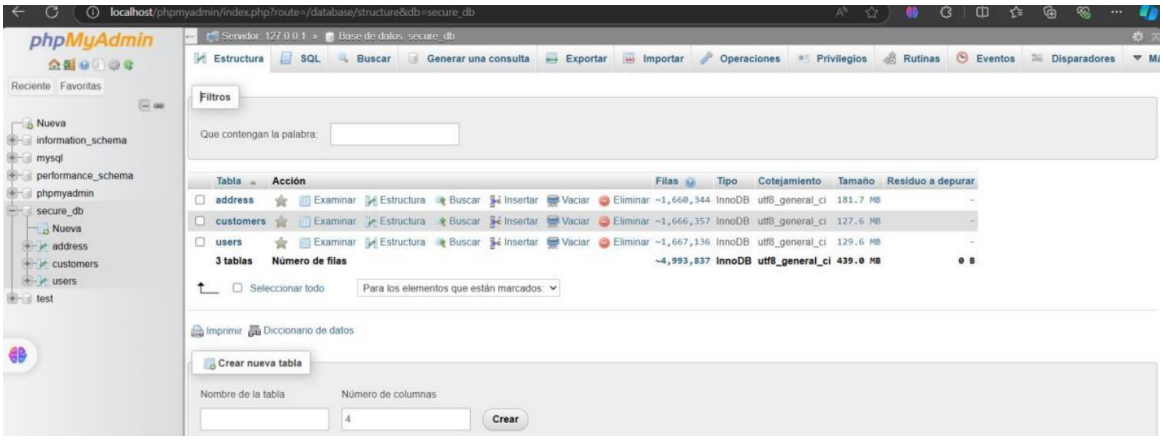
```
if __name__ == "__main__":
    perform_sql_injection()
```

Poateriormente aquí es donde muestran desde la tabla por consola:

```
(1682387, 'gregg23@olsen-harmon.com', 'default_password', 'cCBa62dBe5e0966')
(1682388, 'candice78@benjamin-cisneros.net', 'default_password', 'BAd4D57162eD079')
(1682389, 'lutzmegan@roberson.com', 'default_password', 'd04A2CDfc5805b8')
(1682390, 'walter19@eaton.com', 'default_password', '0976f72B9E76f80')
(1682391, 'alee@villanueva.com', 'default_password', '6A5DB6BB8D33DA4')
(1682392, 'kle@sawyer-villa.net', 'default_password', '6DdB3FB0CC8B5ed')
(1682393, 'rushariana@ramos.info', 'default_password', '12858C5BF27D1DB')
(1682394, 'ealvarez@sanford.com', 'default_password', '798C531bD660e2A')
(1682395, 'sethbuckley@hawkins.info', 'default_password', '2e8BcC495de41bD')
(1682396, 'lmoran@alvarez.info', 'default_password', 'a7eFAf58c6ddad7')
(1682397, 'collinbanks@arellano.biz', 'default_password', '9Eebde8d724bd05')
```

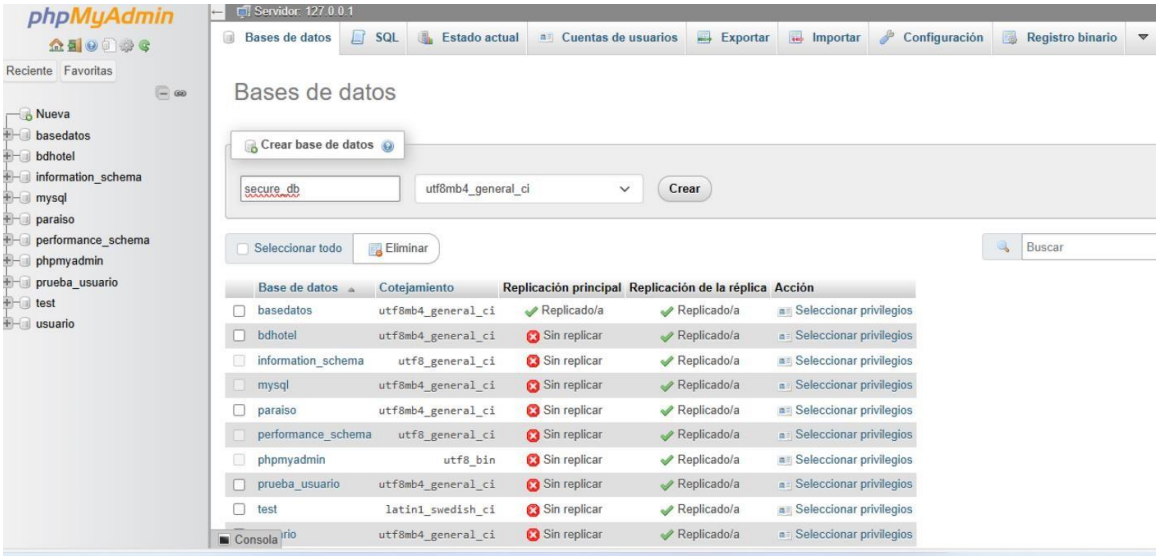
4. Crea un backup de la base de datos `secure_db` y restaura la base de datos en un servidor diferente.

Principalmente se realiza lo que es las tablas en la base de datos donde se realizara el backup, como se puede observar a continuación en la siguiente imagen:



Después nos dirigimos a la opción de exportar como se puede observar a continuación en la siguiente imagen:

Después lo probamos en lo que en otra computadora para poder ver la base de datos, y posteriormente le damos en crear, como se puede ver a continuación en la siguiente imagen:



Y finalmente vemos que se creo todo como se puede ver a continuación en la siguiente imagen:

INVESTIGACION:

conceptos de SQL Inyección y cómo se pueden prevenir

Introducción a SQL

El SQL (Structured Query Language) es un lenguaje de consulta estructurado desarrollado en la década de 1970. Se ha convertido en el estándar para la gestión de bases de datos, permitiendo a los desarrolladores realizar consultas para acceder y manipular datos. Cada base de datos puede tener su propio conjunto de normas SQL, lo que significa que no se puede simplemente copiar y pegar código de una base de datos a otra.

¿Qué es la Inyección de SQL?

La inyección de SQL(SQLI) es un tipo de ataque cibernético en el que un hacker introduce código SQL malicioso en un sitio web vulnerable. Este ataque se produce cuando el desarrollador no implementa un **saneamiento de entrada** adecuado, permitiendo que el código malicioso se ejecute en la base de datos como si fuera una consulta legítima.

Cómo Ocurre un Ataque de Inyección de SQL

Los ataques de inyección de SQL pueden llevarse a cabo de varias maneras:

-Introducción de Datos del Usuario: Los hackers pueden inyectar código SQL a través de formularios de entrada, comentarios o reseñas en un sitio web.

- Modificación de Cookies: Los ciberdelincuentes pueden manipular cookies para enviar código SQL al servidor.

- Variables de Servidor: Los hackers pueden insertar código SQL en las solicitudes del navegador que no son adecuadamente saneadas.

- Herramientas de Hackeo Automáticas: Existen herramientas como SQLMAP que pueden detectar y aprovechar vulnerabilidades en sitios web.

Consecuencias de los Ataques de Inyección de SQL

Los ataques de inyección de SQL pueden tener efectos devastadores tanto para los usuarios como para las empresas:

Efectos en los Usuarios

- Pérdida de Dinero: Los hackers pueden transferir dinero desde cuentas de usuarios en sitios financieros.

- Robo de Identidad: Los datos robados pueden ser vendidos en la red oscura, facilitando el robo de identidad y fraudes.

Efectos en las Empresas

- Sabotaje: Los hackers pueden borrar bases de datos o destruir sitios web.

- Robo de Datos: Información confidencial, como secretos comerciales y datos de clientes, puede ser robada.

- Filtraciones de Seguridad: Un ataque exitoso puede permitir a los hackers acceder a otras partes de la red interna de la empresa.

- Pérdida de Reputación: La confianza del cliente puede verse gravemente afectada tras un ataque.

Inyección de SQL mediante variables de servidor

Al introducir la URL de un sitio web en el navegador, tiene lugar una rápida secuencia de comunicaciones cuya finalidad es ofrecer el sitio al usuario. Dentro de este proceso, el navegador solicita una lista de datos denominada «variables de servidor» que sirve para que el sitio se renderice correctamente.

Costos de los Ataques de Inyección de SQL

Los costos de remediación de un ataque de inyección de SQL pueden ser significativos. Un estudio de 2014 reveló que el costo de subsanar un ataque leve puede rondar los 200,000 dólares. En casos más graves, como el ataque a la Marina de EE. UU., los costos superaron los 500,000 dólares.

Prevención de Inyección de SQL

Aunque los usuarios no pueden prevenir directamente un ataque de inyección de SQL, pueden tomar medidas para protegerse:

Hábitos de Navegación Segura

- No Proporcionar Información Personal: Solo ingresar datos en sitios web de confianza.

- Mantenerse Informado: Estar al tanto de las noticias sobre seguridad tecnológica y actuar rápidamente si se informa de un ataque.

- Crear Contraseñas Seguras: Utilizar contraseñas únicas y robustas para cada cuenta.

- Usar un Administrador de Contraseñas: Esto puede ayudar a gestionar contraseñas y alertar sobre brechas de seguridad.

La inyección de SQL es una amenaza seria que puede tener consecuencias devastadoras tanto para individuos como para empresas. La prevención y el saneamiento adecuado de la entrada son esenciales para protegerse contra estos ataques. Mantenerse informado y adoptar buenas prácticas de seguridad puede ayudar a mitigar los riesgos asociados con la inyección de SQL.

Prevención de Inyección de SQL

Aunque los usuarios no pueden prevenir directamente un ataque de inyección de SQL, pueden tomar medidas para protegerse:

Saneamiento de Entrada: Los desarrolladores deben asegurarse de que todos los datos introducidos por los usuarios sean validados y limpiados adecuadamente.

Uso de Herramientas de Seguridad: Implementar herramientas como SQLMAP para detectar vulnerabilidades y corregirlas.

Educación y Conciencia:

No Proporcionar Información Personal: Solo ingresar datos en sitios web de confianza.

Mantenerse Informado: Estar al tanto de las noticias sobre seguridad tecnológica y actuar rápidamente si se informa de un ataque.

Crear Contraseñas Seguras: Utilizar contraseñas únicas y robustas para cada cuenta.

Utilizar un Administrador de Contraseñas: Esto puede ayudar a gestionar contraseñas y alertar sobre brechas de seguridad.

Conclusión

Conceptos de Bases de Datos Seguras y cómo se pueden Implementar

La seguridad de las bases de datos es un aspecto crítico en la gestión de la información, ya que las brechas de seguridad pueden tener consecuencias devastadoras tanto para individuos como para organizaciones. Este documento explora las principales amenazas a la seguridad de las bases de datos, sus causas y las estrategias para prevenir ataques.

Tipos de Amenazas a la Seguridad de Bases de Datos

Amenazas Internas

Las amenazas internas son una de las causas más comunes de brechas de seguridad. Estas pueden surgir de:

- Infiltrados Malintencionados: Empleados o contratistas que buscan causar daño deliberadamente.

- Infiltrados Negligentes: Personas que cometen errores que pueden hacer que la base de datos sea vulnerable.

- Infiltrados Externos: Extraños que obtienen credenciales a través de técnicas como el phishing.

La falta de control sobre quién tiene acceso a la base de datos puede aumentar el riesgo de estas amenazas.

Implementación de Bases de Datos Seguras

Cifrado de Datos:

Utilizar cifrado para proteger datos sensibles almacenados en la base de datos.

Implementar cifrado en tránsito utilizando protocolos como TLS/SSL.

Autenticación Fuerte:

Implementar autenticación multifactor (MFA) para el acceso a la base de datos.

Utilizar métodos de autenticación actualizados, como OAuth o SAML.

Control de Acceso Basado en Roles (RBAC):

Asignar roles específicos a usuarios y otorgar permisos según las necesidades del trabajo.

Revisar y auditar regularmente los permisos de acceso.

Seguridad en la Configuración:

Asegurarse de que la base de datos esté configurada de manera segura (e.g., desactivar funciones innecesarias, cambiar contraseñas predeterminadas).

Limitar la conectividad de la base de datos a través de firewalls o redes privadas virtuales (VPN).

Auditoría y Monitoreo:

Implementar registros de auditoría para realizar un seguimiento de las acciones de los usuarios en la base de datos.

Usar herramientas de monitoreo para detectar actividades sospechosas.

Actualizaciones y Parches:

Mantener el software de la base de datos actualizado con los últimos parches de seguridad.

Realizar pruebas de penetración y análisis de vulnerabilidades regularmente.

Respaldo y Recuperación:

Establecer un plan de respaldo regular y pruebas de recuperación de datos.

Almacenar copias de seguridad en ubicaciones seguras y verificar que sean accesibles y utilizables en caso de emergencia.

Segmentación de Datos:

Clasificar los datos en función de su sensibilidad y aplicar medidas de seguridad adicionales a aquellos que requieren mayor protección.

El error humano es responsable de casi el 49% de todas las brechas de seguridad. Esto incluye:

- Uso de contraseñas débiles.

- Compartición de contraseñas.

- Comportamientos imprudentes que pueden comprometer la seguridad.

Explotación de Vulnerabilidades de Software

Los hackers buscan constantemente vulnerabilidades en el software de gestión de bases de datos. Los proveedores de software emiten parches de seguridad, pero no aplicarlos a tiempo puede aumentar la exposición a ataques.

Ataques de Inyección SQL/NoSQL

La inyección de SQL es un ataque que permite a los hackers insertar código SQL malicioso en consultas de bases de datos. Esto puede ocurrir si las aplicaciones web no siguen prácticas de codificación seguras. Las organizaciones que no realizan pruebas de vulnerabilidad regularmente están en riesgo.

Explotaciones de Desbordamiento de Búfer

El desbordamiento de búfer ocurre cuando un proceso intenta escribir más datos en un bloque de memoria de lo que puede contener. Esto puede ser utilizado por atacantes para lanzar otros tipos de ataques.

Malware

El malware está diseñado para explotar vulnerabilidades y causar daños a la base de datos. Puede infiltrarse a través de dispositivos conectados a la red.

Ataques a Copias de Seguridad

Las copias de seguridad que no están protegidas adecuadamente son vulnerables a ataques. Es crucial aplicar los mismos controles de seguridad que se utilizan para proteger la base de datos principal.

Factores Agravantes

- Crecimiento de Datos: La cantidad de datos que las organizaciones manejan está aumentando exponencialmente, lo que requiere herramientas de seguridad escalables.

- Complejidad de Infraestructura: La transición a arquitecturas de nube híbrida o multinube complica la gestión de la seguridad.

- Requisitos Regulatorios: La creciente complejidad de la conformidad normativa dificulta el cumplimiento de todos los mandatos.

-Escasez de Habilidades en Ciberseguridad: La falta de expertos en ciberseguridad es un desafío significativo, con predicciones de hasta 8 millones de puestos sin cubrir.

Ataques de Denegación de Servicio (DoS/DDoS)

En un ataque DoS, el atacante inunda el servidor de la base de datos con solicitudes, lo que puede hacer que el servidor se vuelva inestable o se bloquee. En un ataque DDoS, el ataque proviene de múltiples servidores, lo que dificulta su contención.

Consecuencias de los Ataques de Inyección de SQL

Los ataques de inyección de SQL pueden tener efectos devastadores:

Efectos en las Personas

- Pérdida de Dinero: Los hackers pueden transferir fondos desde cuentas bancarias.

- Robo de Identidad: La información personal puede ser robada y vendida en la red oscura.

Efectos en las Empresas

- Sabotaje: Los hackers pueden eliminar datos o destruir la infraestructura del sitio web.

- Robo de Datos: Información confidencial puede ser sustraída.

- Filtraciones de Seguridad: Un ataque exitoso puede permitir a los atacantes acceder a otras partes de la red interna.

- Pérdida de Reputación: La confianza del cliente puede verse gravemente afectada.

Prevención de Ataques de Inyección de SQL

Para prevenir ataques de inyección de SQL, es fundamental:

1. Saneamiento de Entrada: Validar y limpiar todos los datos de entrada.

2.Control de Acceso: Limitar el acceso a la base de datos solo a personal autorizado.

3.Aplicación de Parches: Mantener el software actualizado con los últimos parches de seguridad.

4. Capacitación del Personal: Educar a los empleados sobre las mejores prácticas de seguridad.

5.Monitoreo y Auditoría: Implementar sistemas de monitoreo para detectar actividades sospechosas.

La seguridad de las bases de datos es un componente esencial de la gestión de la información en las organizaciones. Implementar medidas de seguridad adecuadas y educar a los empleados sobre las mejores prácticas puede ayudar a mitigar los riesgos asociados con los ataques de inyección de SQL y otras amenazas.

CONCLUSION:

Con esto llegamos a la conclusión de la práctica que se realizó donde quedo como entendido que la implementación de medidas de seguridad en las bases de datos es esencial para garantizar la protección de la información en un entorno cada vez más expuesto a amenazas internas y externas. A lo largo de esta práctica, se ha evidenciado la importancia de aplicar estrategias como el control de acceso, la autenticación robusta, el cifrado y el monitoreo continuo, que son fundamentales para mantener la confidencialidad, integridad y disponibilidad de los datos. Comprender las vulnerabilidades más comunes, como la inyección SQL y los errores en la configuración de permisos, nos permite adoptar enfoques preventivos y correctivos para minimizar los riesgos. A través del uso adecuado de estas técnicas, las organizaciones pueden fortalecer su infraestructura de bases de datos, reducir la superficie de ataque y cumplir con normativas de seguridad de datos, lo que es crucial para generar confianza y asegurar la continuidad del negocio. En resumen, la seguridad en bases de datos no es un esfuerzo aislado, sino un proceso continuo que requiere una actualización constante de herramientas y prácticas, así como una conciencia clara de los riesgos emergentes en el ámbito de la seguridad informática.

BIBLIOGRAFIAS:

Avast. (n.d.). ¿Qué es la inyección de SQL? Recuperado de https://www.avast.com/es-es/c-sql-injection

IBM. (n.d.). Seguridad de Bases de Datos. Recuperado el [fecha de consulta], de https://www.ibm.com/mx-es/topics/database-security