

INGENIERÍA EN SISTEMAS COMPUTACIONALES.

SEGURIDAD Y VIRTUALIZACION.



PRACTICA 3: BASES DE DATOS SEGURAS.

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 17 DE SEPTIEMBRE DE 2024.

ÍNDICE DE ILUSTRACIONES.

Ilustración 1: Código para crear la BD.....	4
Ilustración 2: Conexión a la base de datos.....	5
Ilustración 3: Script para insertar datos (parte 1).....	6
Ilustración 4: Script para insertar datos (parte 2).....	7
Ilustración 5: Script para insertar datos (parte 3).....	7
Ilustración 6: Inserción de dos datos del archivo customers-2000000.csv a la BD.	8
Ilustración 7: Datos insertados Parte1.....	8
Ilustración 8: Datos insertados Parte2.....	9
Ilustración 9: : Permiso de lectura en la tabla `customers`	9
Ilustración 10: Permisos de lectura y escritura en la tabla `address`	9
Ilustración 11: Permisos de lectura, escritura y eliminación en la tabla `users`	10
Ilustración 12: Script para insertar Datos.....	10
Ilustración 13: Script para insertar Datos(2).	11
Ilustración 14: Ejecución en consola del Script.	11
Ilustración 15: Tabla de datos insertados.	12
Ilustración 16: Tabla de datos en XAMPP MySQL.	12
Ilustración 17: Script de Inyección (1).....	13
Ilustración 18: Script de Inyección (2).....	13
Ilustración 19: Datos extraídos.	14
Ilustración 20: Descargar SQL Dump Splittr.....	15
Ilustración 21: Mensaje de Bienvenida del Software.	15
Ilustración 22: Base de datos Secure_db	16
Ilustración 23: Ruta para guardar los archivos.	16
Ilustración 24: Base de datos nueva "respaldo_secure_db".....	17
Ilustración 25: Importar archivos.	17
Ilustración 26: Datos de tabla Customers.....	18
Ilustración 27: Datos de Tabla Users.....	18

DESARROLLO DE LA PRACTICA.

- **Crea una base de datos en MySQL con una BD que contiene los siguientes campos en tres tablas diferentes:**

```
CREATE DATABASE IF NOT EXISTS `secure_db` DEFAULT CHARACTER SET  
utf8 COLLATE utf8_general_ci;
```

```
USE `secure_db`;
```

```
CREATE TABLE IF NOT EXISTS `users` (  
    `id` INT NOT NULL AUTO_INCREMENT,  
    `email` VARCHAR(45) NOT NULL,  
    `password` VARCHAR(45) NOT NULL,  
    `customer_id` VARCHAR(45) NOT NULL REFERENCES  
customers(customer_id),  
    PRIMARY KEY (`id`))
```

```
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `address` (  
    `id` INT NOT NULL AUTO_INCREMENT,  
    `company` VARCHAR(45) NOT NULL,  
    `city` VARCHAR(45) NOT NULL,  
    `country` VARCHAR(45) NOT NULL,  
    `phone_1` VARCHAR(45) NOT NULL,  
    `phone_2` VARCHAR(45) NOT NULL,  
    `customer_id` VARCHAR(45) NOT NULL REFERENCES  
customers(customer_id),
```

PRIMARY KEY (`id`))

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `customers` (

 `id` INT NOT NULL AUTO_INCREMENT,

 `customer_id` VARCHAR(45) NOT NULL,

 `first_name` VARCHAR(45) NOT NULL,

 `last_name` VARCHAR(45) NOT NULL,

 `subscription_date` DATE NOT NULL,

 `website` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

1. Para realizar este primer paso lo primero que se realizó fue ejecutar el código que se presenta anteriormente en XAMPP MySQL para crear la base de datos `secure_db` y las tablas necesarias:

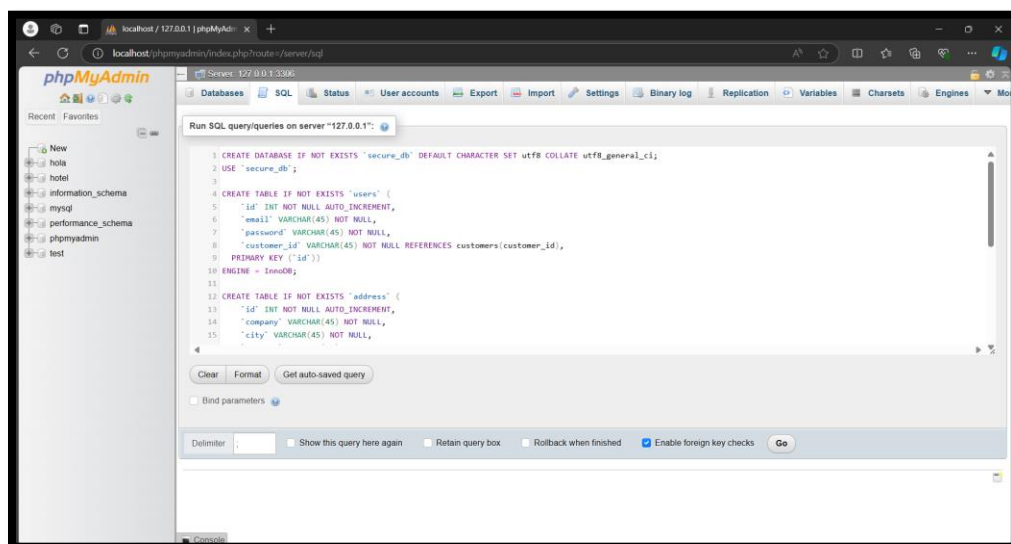


Ilustración 1: Código para crear la BD.

2. Una vez ejecutado la acción podemos apreciar que se muestra la base de datos con sus respectivas tablas.

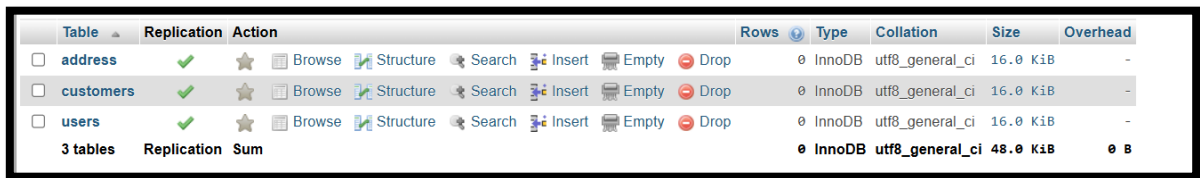


Table	Replication	Action	Rows	Type	Collation	Size	Overhead
address	OK	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	16.0 KiB	-
customers	OK	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	16.0 KiB	-
users	OK	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	16.0 KiB	-
3 tables	Replication	Sum	0	InnoDB	utf8_general_ci	48.0 KiB	0 B

Ilustración 2: Tablas de la base de datos.

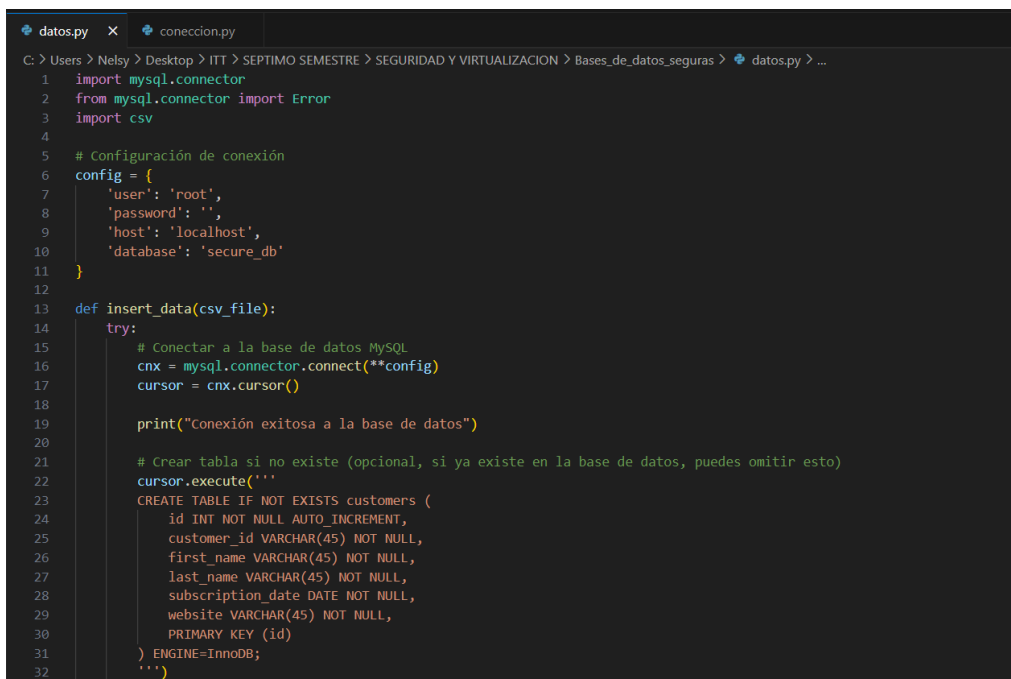
3. Crea un script en Python que permita insertar los datos del archivo `customers-2000000.csv`

1. Para insertar los datos desde el archivo, el primer paso es establecer una conexión con la base de datos en XAMPP. Para ello, utilizamos Python y el lenguaje de programación correspondiente para crear un script que se encargue de la inserción de datos. Este script se desarrolla y edita utilizando herramientas como Visual Studio Code.

```
1 import mysql.connector
2 from mysql.connector import Error
3
4 # Configuración de conexión
5 config = {
6     'user': 'root',
7     'password': '',
8     'host': 'localhost', # o la IP del servidor MySQL
9     'database': 'secure_db'
10 }
11
12 try:
13     # Intentar la conexión
14     cnx = mysql.connector.connect(**config)
15
16     # Verificar si la conexión fue exitosa
17     if cnx.is_connected():
18         print("Conexión exitosa a la base de datos")
19
20     # Aquí podrías ejecutar tus consultas
21
22 except Error as e:
23     print(f"Error al conectarse a la base de datos: {e}")
24
25 finally:
26     # Cerrar la conexión si está abierta
27     if cnx.is_connected():
28         cnx.close()
29     print("Conexión cerrada")
```

Ilustración 3: Conexión a la base de datos.

2. Posteriormente creamos el script para que inserte los datos necesarios: Este script en Python se conecta a la base de datos MySQL y realiza la inserción de datos desde un archivo CSV en la tabla customers. Primero, establece la conexión a la base de datos secure_db en el servidor local con las credenciales especificadas. Luego, verifica si la tabla customers existe y, en caso contrario, la crea. Después, abre el archivo CSV, lee cada fila e inserta los datos correspondientes en la tabla. Durante este proceso, el script maneja posibles errores en la inserción de datos y confirma los cambios en la base de datos. Finalmente, cierra la conexión a la base de datos y el cursor para liberar los recursos.



```
1 import mysql.connector
2 from mysql.connector import Error
3 import csv
4
5 # Configuración de conexión
6 config = {
7     'user': 'root',
8     'password': '',
9     'host': 'localhost',
10    'database': 'secure_db'
11 }
12
13 def insert_data(csv_file):
14     try:
15         # Conectar a la base de datos MySQL
16         cnx = mysql.connector.connect(**config)
17         cursor = cnx.cursor()
18
19         print("Conexión exitosa a la base de datos")
20
21         # Crear tabla si no existe (opcional, si ya existe en la base de datos, puedes omitir esto)
22         cursor.execute('''
23         CREATE TABLE IF NOT EXISTS customers (
24             id INT NOT NULL AUTO_INCREMENT,
25             customer_id VARCHAR(45) NOT NULL,
26             first_name VARCHAR(45) NOT NULL,
27             last_name VARCHAR(45) NOT NULL,
28             subscription_date DATE NOT NULL,
29             website VARCHAR(45) NOT NULL,
30             PRIMARY KEY (id)
31         ) ENGINE=InnoDB;
32         ''')
```

Ilustración 4: Script para insertar datos (parte 1).

```
datos.py x conexion.py
C:\Users\Nelsy\Desktop> ITT > SEPTIMO SEMESTRE > SEGURIDAD Y VIRTUALIZACION > Bases_de_datos_seguras > datos.py > ...
13 def insert_data(csv_file):
31     ) ENGINE=InnoDB;
32     '''
33
34     print("Tabla `customers` verificada o creada")
35
36     # Insertar datos desde el archivo CSV
37     with open(csv_file, newline='', encoding='utf-8') as file:
38         reader = csv.DictReader(file)
39         for row in reader:
40             try:
41                 cursor.execute('''
42                     INSERT INTO customers (customer_id, first_name, last_name, subscription_date, website)
43                     VALUES (%s, %s, %s, %s, %s)
44                     ''', (
45                         row['Customer Id'],
46                         row['First Name'],
47                         row['Last Name'],
48                         row['Subscription Date'],
49                         row['Website']
50                     ))
51                 print(f"Fila insertada: {row}") # Mensaje de depuración
52             except Error as e:
53                 print(f"Error al insertar fila: {e}")
54
55             # Confirmar los cambios
56             cnx.commit()
57             print("Datos insertados correctamente")
58
59     except Error as e:
60         print(f"Error al conectar o realizar operaciones: {e}")
61
62     finally:
```

Ilustración 5: Script para insertar datos (parte 2).

```
except Error as e:
    print(f"Error al conectar o realizar operaciones: {e}")

finally:
    # Cerrar la conexión
    if cnx.is_connected():
        cursor.close()
        cnx.close()
        print("Conexión cerrada")

# Llamar a la función con el archivo CSV
insert_data('C:/Users/Nelsy/Desktop/ITT/SEPTIMO SEMESTRE/SEGURIDAD Y VIRTUALIZACION/Bases_de_datos_seguras/customers-2000000.csv')
```

Ilustración 6: Script para insertar datos (parte 3).

- Una vez completado este paso, ejecutamos el script. Si el código se ejecuta correctamente, insertará los datos del archivo customers-2000000.csv en secure_db.

```

ail': 'ross22@stanley.com', 'Subscription Date': '2020-10-01', 'Website': 'https://www.vincent-choi.info/')
Fila insertada: {'Index': '1999992', 'Customer Id': 'ad5A9c21B2cc9Bf', 'First Name': 'Shari', 'Last Name': 'Mullen', 'Company': 'Perkin
s PLC', 'City': 'Port Tanner', 'Country': 'Netherlands', 'Phone 1': '450.175.7407', 'Phone 2': '001-887-564-4863x9010', 'Email': 'diane
donovan@andrews.org', 'Subscription Date': '2020-10-01', 'Website': 'https://www.vaughan.biz/')
Fila insertada: {'Index': '1999993', 'Customer Id': 'E8aE7bEFBBE83b4', 'First Name': 'Jody', 'Last Name': 'Gillespie', 'Company': 'Mcne
il, Reid and Kim', 'City': 'West Willie', 'Country': 'Denmark', 'Phone 1': '41-622-519-4135', 'Phone 2': '41-619-306-1953x50720', 'Emai
l': 'dustingolden@kirk.com', 'Subscription Date': '2021-04-22', 'Website': 'https://www.burch-mullen.biz/')
Fila insertada: {'Index': '1999994', 'Customer Id': '51BcFD3D3e57Fcb', 'First Name': 'Summer', 'Last Name': 'Bowen', 'Company': 'Case G
roup', 'City': 'Veronicaborough', 'Country': 'Brazil', 'Phone 1': '645.904.8275x1625', 'Phone 2': '(216)349-9829x43604', 'Email': 'ubar
ker@duffy-carroll.net', 'Subscription Date': '2020-06-13', 'Website': 'http://www.walton-donovan.org/')
Fila insertada: {'Index': '1999995', 'Customer Id': 'a7d56b5031BbAfc', 'First Name': 'Glenda', 'Last Name': 'Andrews', 'Company': 'West
and Sons', 'City': 'North Blake', 'Country': 'Canada', 'Phone 1': '940-853-4463x20580', 'Phone 2': '+1-150-376-5610', 'Email': 'bianca
bartlett@bruce.net', 'Subscription Date': '2020-07-17', 'Website': 'https://www.davenport-rios.com/')
Fila insertada: {'Index': '1999996', 'Customer Id': '8cdADE4D53EF36B', 'First Name': 'Christian', 'Last Name': 'Sweeney', 'Company': 'P
atterson Group', 'City': 'Kristiside', 'Country': 'Uruguay', 'Phone 1': '884-318-2264x60624', 'Phone 2': '642.871.5952x89561', 'Email':
'clintonhardy@sweeney.net', 'Subscription Date': '2021-11-07', 'Website': 'http://rosario.com/')
atterson Group', 'City': 'Kristiside', 'Country': 'Uruguay', 'Phone 1': '884-318-2264x60624', 'Phone 2': '642.871.5952x89561', 'Email':
atterson Group', 'City': 'Kristiside', 'Country': 'Uruguay', 'Phone 1': '884-318-2264x60624', 'Phone 2': '642.871.5952x89561', 'Email':
'clintonhardy@sweeney.net', 'Subscription Date': '2021-11-07', 'Website': 'http://rosario.com/')
Fila insertada: {'Index': '1999997', 'Customer Id': '5c2D38BBc412ccd', 'First Name': 'Evan', 'Last Name': 'Morrow', 'Company': 'Huerta,
Murphy and Rowland', 'City': 'Glennton', 'Country': 'Romania', 'Phone 1': '(856)972-5239x6164', 'Phone 2': '471-254-8874x1575', 'Email':
'philippotter@anthony-chapman.com', 'Subscription Date': '2020-04-26', 'Website': 'https://lawrence-kaufman.com/')
Fila insertada: {'Index': '1999998', 'Customer Id': 'E57d2b9a88EcD88', 'First Name': 'Deborah', 'Last Name': 'Zimmerman', 'Company': 'B
lancharde-Reynolds', 'City': 'East Ashleytown', 'Country': 'Saint Lucia', 'Phone 1': '(920)325-1228', 'Phone 2': '(855)827-3943x1016', '
Email': 'micheal50@woodard.com', 'Subscription Date': '2020-12-16', 'Website': 'http://www.garrison.info/')
Fila insertada: {'Index': '1999999', 'Customer Id': 'ae4EA5B4d9A84e', 'First Name': 'Frederick', 'Last Name': 'Villanueva', 'Company':
'Buchanan-Daugherty', 'City': 'North Russellhaven', 'Country': 'Oman', 'Phone 1': '(546)217-9813x78237', 'Phone 2': '666-409-8068x561',
'Email': 'cuevaskrista@herman-lin.com', 'Subscription Date': '2021-04-25', 'Website': 'https://buckley-potts.net/')
Fila insertada: {'Index': '2000000', 'Customer Id': '68d72e3af39fADF', 'First Name': 'Tami', 'Last Name': 'Hopkins', 'Company': 'Rocha,
Barnes and Wood', 'City': 'Valerieberg', 'Country': 'Macedonia', 'Phone 1': '+1-558-089-7618', 'Phone 2': '001-592-639-4940x79440', 'E
mail': 'smithyesenia@dean.biz', 'Subscription Date': '2021-10-17', 'Website': 'http://www.thornton.com/')
Datos insertados correctamente
Conexión cerrada
PS C:\Users\Welsy>

```

Ilustración 7: Inserción de dos datos del archivo customers-2000000.csv a la BD.

- Verificamos que los datos se hallan insertado correctamente en la base de datos, específicamente en la tabla customers.

id	customer_id	first_name	last_name	subscription_date	website
1781776	4962kcbE6Bfee6D	Pam	Sparks	2020-11-29	https://nelson.com/
1781779	9b12Ae76kdbC5bE	Gina	Rocha	2021-01-03	https://pineda-rogers.biz/
1781780	39edFd2F60C85BC	Kristie	Greer	2021-06-20	https://mckinney.com/
1781781	Fa42AE6a9aD39cE	Arthur	Fields	2020-02-13	https://dominguez.biz/
1781782	F5702Edae925F1D	Michelle	Blevins	2020-10-20	http://munillo-ryan.com/
1781783	9FbId34c3f30d9	Greg	Rivers	2020-02-19	https://www.richard.biz/
1781784	D31D38D576a81Bf	Brian	Brandt	2020-03-31	https://www.washington.org/
1781785	3C3B8bee16BcC5F	Dennis	Pacheco	2021-03-20	http://www.armstrong-golden.com/
1781786	7C1C93522Bd5e4B	Malik	Rivers	2020-01-16	https://watson.com/
1781787	bABC8cBfe68792	Hayden	Riddle	2022-05-18	https://www.farley.net/
1781788	534a2B5F8Abe5d	Edward	Wood	2020-05-22	http://alvarez.com/
1781789	4baa944fAEbe9d	Julie	Stein	2020-01-17	https://carlson-johnston.com/
1781790	Fc2c8D2BE1AEIDb	Kristina	Andrade	2020-09-11	https://foley.com/
1781791	9468BBc926AaAB3	Zoe	Hansen	2021-10-09	https://franco-galloway.com/
1781792	BE96AcDDCc77daD	Joseph	Bennett	2020-04-29	https://howard-boyer.net/
1781793	A24E162EB6d97B	Vickie	Ferguson	2021-04-16	https://ashley-oneill.info/
1781794	f1a979d0Ed2dB14	Dillon	Frye	2021-11-29	http://www.young.com/
1781795	dCBc761B59e4EFB	Grace	Logan	2020-02-11	https://www.knapp.net/

Ilustración 8: Datos insertados Parte1.

<input type="checkbox"/>				1781797	8Ace2d52a65faA6	Philip	Andersen	2020-02-04	http://patton-pineda.com/
<input type="checkbox"/>				1781798	479BDefDBe92D35	Meghan	Price	2020-03-08	https://downs-meyers.com/
<input type="checkbox"/>				1781799	A1505BF376CC5Ed	Aimee	Brooks	2022-03-28	http://solis.org/
<input type="checkbox"/>				1781800	eeAAb881488E030	Sheryl	Stein	2021-09-23	http://black.com/
<input type="checkbox"/>				1781801	5b04Fa517bF3a85	Tony	Sanford	2022-02-22	https://mason.info/
<input type="checkbox"/>				1781802	dbCDD2A070edCb4	Johnny	Mcdaniel	2021-07-20	https://morales-watson.org/
<input type="checkbox"/>				1781803	34BEBcc82bC50Dd	Dominic	Wiggins	2020-04-17	http://barker-padilla.net/
<input type="checkbox"/>				1781804	B8f38cCfdCc2C1A	Sue	Owens	2020-08-25	http://archer.com/
<input type="checkbox"/>				1781805	a24eB840950dac7	Mackenzie	Leonard	2022-05-02	http://www.pacheco.net/
<input type="checkbox"/>				1781806	13D7dF9dccCea1e	Eugene	Haas	2022-03-18	http://www.maldonado.com/
<input type="checkbox"/>				1781807	dD1E3AECf73C4C3	Raven	Cohen	2022-01-24	http://andersen-lyons.com/
<input type="checkbox"/>				1781808	8cfD28baAdDa327	Mario	Poole	2022-03-18	http://baxter.org/
<input type="checkbox"/>				1781809	A4DCa561daD6Bee	Duane	Hickman	2020-04-25	http://www.clay.org/
<input type="checkbox"/>				1781810	B0D6fcC44d9D730	Jose	Herrera	2020-03-10	https://www.jacobson.com/
<input type="checkbox"/>				1781811	feE562AE4cc1C37	Rachael	Hubbard	2021-08-03	http://www.silva-singleton.com/
<input type="checkbox"/>				1781812	7Ae7E7f9c159dCA	Gabriela	Kane	2021-01-20	https://santos.com/
<input type="checkbox"/>				1781813	106abC5fa1774A7	Zachary	Patterson	2021-08-31	http://stevens.com/
<input type="checkbox"/>				1781814	B47F8FFD6D40392	Roberto	Fox	2020-06-29	http://gregory-wolfe.com/
<input type="checkbox"/>				1781815	e39fc890ABaFB99	Shelby	Hayes	2020-07-03	https://www.bridges.net/

Ilustración 9: Datos insertados Parte2.

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

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

```
1 CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';
2 GRANT SELECT ON secure_db.customers TO 'user1'@'localhost';
```

Ilustración 10: : Permiso de lectura en la tabla `customers`

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

```
1 CREATE USER 'usuario2'@'localhost' IDENTIFIED BY 'contraseña123';
2 GRANT SELECT, INSERT, UPDATE ON secure_db.address TO 'usuario2'@'localhost';
```

Ilustración 11: Permisos de lectura y escritura en la tabla `address`

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

```

1 CREATE USER 'user3'@'localhost' IDENTIFIED BY 'password3';
2 GRANT SELECT, INSERT, UPDATE, DELETE ON secure_db.users TO 'user3'@'localhost';
3 FLUSH PRIVILEGES;
4

```

Ilustración 12: Permisos de lectura, escritura y eliminación en la tabla `users`

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.

1. En las siguientes capturas de pantalla se muestra un script que permite insertar datos en la tabla users que cuenta con los datos id, email, password, customer_id y posteriormente muestra los datos insertados en pantalla.

```

C:\Users\Nelsy\Desktop> ITT > SEPTIMO SEMESTRE > SEGURIDAD Y VIRTUALIZACION > Bases_de_datos_seguras >
1 import mysql.connector
2 from mysql.connector import Error
3
4 # Configuración de conexión
5 config = {
6     'user': 'root',
7     'password': '',
8     'host': 'localhost',
9     'database': 'secure_db',
10    'connection_timeout': 300
11 }
12
13 def insert_data():
14     try:
15         # Conectar a la base de datos MySQL
16         cnx = mysql.connector.connect(**config)
17         cursor = cnx.cursor()
18
19         print("Conexión exitosa a la base de datos")
20
21         cursor.execute('''
22         CREATE TABLE IF NOT EXISTS users (
23             id INT NOT NULL AUTO_INCREMENT,
24             email VARCHAR(100) NOT NULL,
25             password VARCHAR(100) NOT NULL,
26             customer_id VARCHAR(45) NOT NULL,
27             PRIMARY KEY (id),
28             FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
29         ) ENGINE=InnoDB;
30         ''')
31
32         print("Tabla `users` verificada o creada")
33
34         # Solicitar datos del usuario
35         while True:
36             email = input("Introduce el email: ")
37             password = input("Introduce el password: ")

```

Ilustración 13: Script para insertar Datos.

```

        customer_id
    ))

    print(f"Datos insertados: Email={email}, Customer ID={customer_id}")

except Error as e:
    print(f"Error al insertar los datos: {e}")
    print(f"SQLSTATE: {e.sqlstate}, Error Code: {e.errno}")

# Preguntar si el usuario quiere insertar otra fila
another = input("¿Deseas insertar otro usuario? (s/n): ")
if another.lower() != 's':
    break

# Reconectar si es necesario antes de confirmar los cambios
cnx.ping(reconnect=True, attempts=3, delay=5)

# Confirmar los cambios
cnx.commit()
print("Todos los datos se han insertado correctamente")

except Error as e:
    print(f"Error al conectar o realizar operaciones: {e}")

finally:
    # Cerrar la conexión
    if cnx.is_connected():
        cursor.close()
        cnx.close()
    print("Conexión cerrada")

# Llamar a la función
insert_data()

```

Ilustración 14: Script para insertar Datos(2).

```

Introduce el email: nelsyortiz0608@gmail.com
Introduce el password: 123456
Introduce el customer_id: 1
Datos insertados: Email=nelsyortiz0608@gmail.com, Customer ID=1
Datos insertados en la base de datos: ID=3, Email=nelsyortiz0608@gmail.com, Password=123456, Customer ID=1
¿Deseas insertar otro usuario? (s/n): s
Introduce el email: lucerozuñiga@12345
Introduce el password: 123456789
Introduce el customer_id: 2
Datos insertados: Email=lucerozuñiga@12345, Customer ID=2
Datos insertados en la base de datos: ID=4, Email=lucerozuñiga@12345, Password=123456789, Customer ID=2
¿Deseas insertar otro usuario? (s/n): s
Introduce el email: arletsalazarnicolas@123
Introduce el password: contraseña1234
Introduce el customer_id: 3
Datos insertados: Email=arletsalazarnicolas@123, Customer ID=3
Datos insertados en la base de datos: ID=5, Email=arletsalazarnicolas@123, Password=contraseña1234, Customer ID=3
¿Deseas insertar otro usuario? (s/n): n
Conexión cerrada

```

Ilustración 15: Ejecución en consola del Script.

```
-----'users'-----
ID=10, Email=nelsyortiz0608@gmail.com, Password=123456, Customer ID=1
ID=11, Email=lucerozuñiga@12345, Password=123456789, Customer ID=1
ID=12, Email=arletsalazarnicolas@123, Password=contraseña123, Customer ID=3

-----
Conexión cerrada
PS C:\Users\Nelsy>
```

Ilustración 16: Tabla de datos insertados.

		id	email	password	customer_id
<input type="checkbox"/>	Edit	10	nelsyortiz0608@gmail.com	123456	1
<input type="checkbox"/>	Edit	11	lucerozuñiga@12345	123456789	1
<input type="checkbox"/>	Edit	12	arletsalazarnicolas@123	contraseña123	3

Ilustración 17: Tabla de datos en XAMPP MySQL.

2. Posteriormente creamos un nuevo script en donde se puede observar una inyección sql que compromete la seguridad de una base de datos. conecta a una base de datos mysql y permite al usuario ingresar un valor que se utiliza en una consulta sql vulnerable. si el usuario introduce un código malicioso como ' or '1'='1, la consulta se transforma en una que devuelve todos los registros de la tabla users, ignorando cualquier filtro de seguridad.

```

Users > Neisy > Desktop > III > SEPTIMO SEMESTRE > SEGURIDAD Y VIRTUALIZACION > Bases_de_datos_seguras > inyeccion.py > ...
1 import mysql.connector
2 from mysql.connector import Error
3
4 # Configuración de conexión
5 config = {
6     'user': 'root',
7     'password': '',
8     'host': 'localhost',
9     'database': 'secure_db',
10    'connection_timeout': 300
11 }
12
13 def sql_injection_example():
14     try:
15         # Conectar a la base de datos MySQL
16         cnx = mysql.connector.connect(**config)
17         cursor = cnx.cursor()
18
19         print("Conexión exitosa a la base de datos")
20
21         # Ejemplo de inyección SQL
22         print("Introduce para realizar una: ' OR '1'='1 ")
23         user_input = input("Intenta una inyección SQL: ")
24
25         # Consulta vulnerable (NO HACER EN PRODUCCIÓN)
26         query = f"SELECT * FROM users WHERE email = '{user_input}' OR 1=1 -- "
27
28         print(f"Ejecutando consulta: {query}") # Para que veas la consulta que se ejecuta
29         cursor.execute(query)
30         rows = cursor.fetchall()
31
32         if rows:
33             print("\nDatos recuperados:")
34             for row in rows:
35                 print(f"ID={row[0]}, Email={row[1]}, Password={row[2]}, Customer ID={row[3]}")
36         else:
37             print("No se encontraron datos.")
38

```

Ilustración 18: Script de Inyección (1).

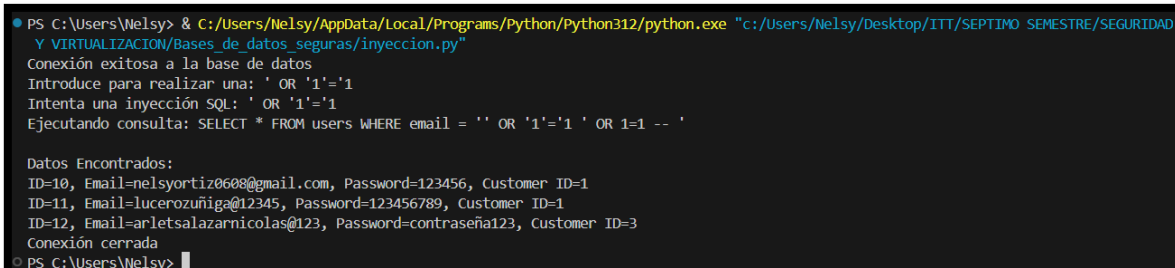
```

39         except Error as e:
40             print(f"Error al conectar o realizar operaciones: {e}")
41
42         finally:
43             # Cerrar la conexión correctamente
44             if cnx.is_connected():
45                 cursor.close()
46                 cnx.close()
47                 print("Conexión cerrada")
48
49 # Llamar a la función
50 sql_injection_example()
51

```

Ilustración 19: Script de Inyección (2).

3. En la siguiente imagen podemos observar la ejecución del código en consola que accede a la base de datos `secure_db`, principalmente a la tabla `users` y extrae los datos.



```
PS C:\Users\Welsy> & C:/Users/Welsy/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Welsy/Desktop/ITT/SEPTIMO SEMESTRE/SEGURIDAD Y VIRTUALIZACION/Bases de datos seguras/inyeccion.py"
Conexión exitosa a la base de datos
Introduce para realizar una: ' OR '1'='1
Intenta una inyección SQL: ' OR '1'='1
Ejecutando consulta: SELECT * FROM users WHERE email = '' OR '1'='1 ' OR 1=1 -- '

Datos Encontrados:
ID=10, Email=nelsyortiz0608@gmail.com, Password=123456, Customer ID=1
ID=11, Email=lucerozuñiga@12345, Password=123456789, Customer ID=1
ID=12, Email=arletsalazarnicolas@123, Password=contraseña123, Customer ID=3
Conexión cerrada
PS C:\Users\Welsy>
```

Ilustración 20: Datos extraídos.

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

1. Para realizar un respaldo de la base de datos en XAMPP, lo primero que debes hacer es exportar la base de datos actual desde la interfaz de phpMyAdmin en XAMPP. Este archivo contendrá una copia completa de la estructura y datos de tu base de datos, y podrás restaurarlo en otro servidor.
2. Una vez completada la copia de seguridad, procederemos a utilizar la herramienta SQL Dump Splitter. Esta aplicación permite dividir una base de datos de gran tamaño en archivos más pequeños, lo que facilita su importación en el nuevo servidor. Para ello, accedemos a la página oficial del software y realizamos la descarga.

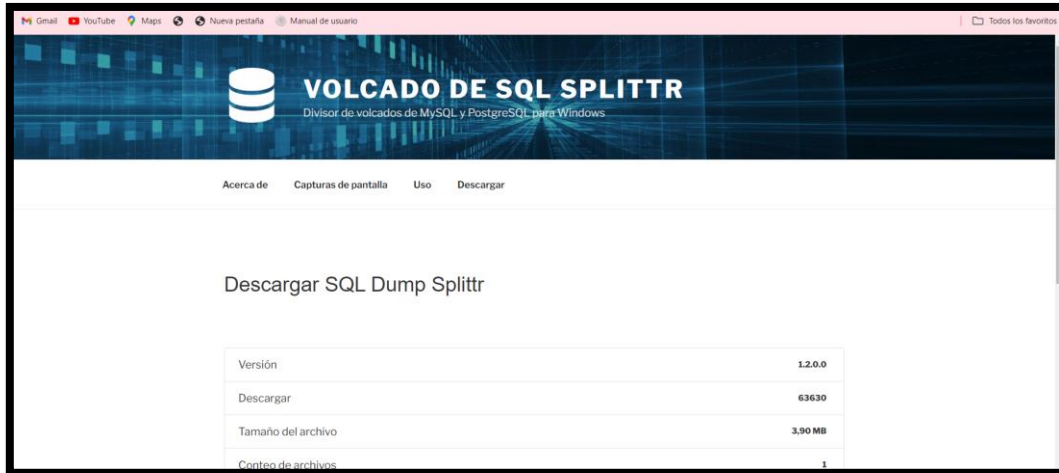


Ilustración 21: Descargar SQL Dump Splitttr.

3. Posteriormente ejecutamos el programa y nos aparecerá el mensaje de bienvenida.

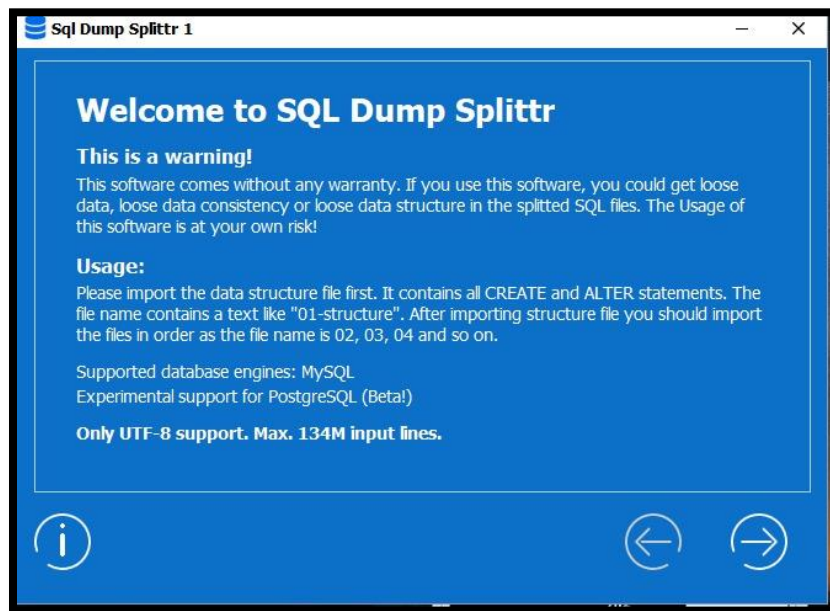


Ilustración 22: Mensaje de Bienvenida del Software.

4. Seleccionamos el archivo que se exportó desde el otro servidor, el cual lleva por nombre Secure_db.sql. A continuación, elegimos MySQL, que es el sistema de gestión de bases de datos que estamos utilizando en este proyecto.

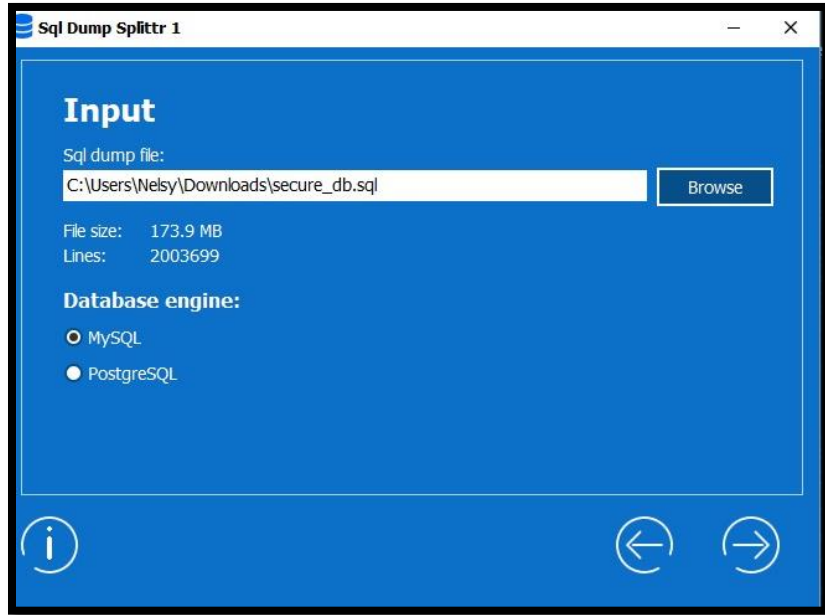


Ilustración 23: Base de datos Secure_db

5. Elegimos la ruta donde se guardarán los archivos de la base de datos y procedemos a guardar los cambios.

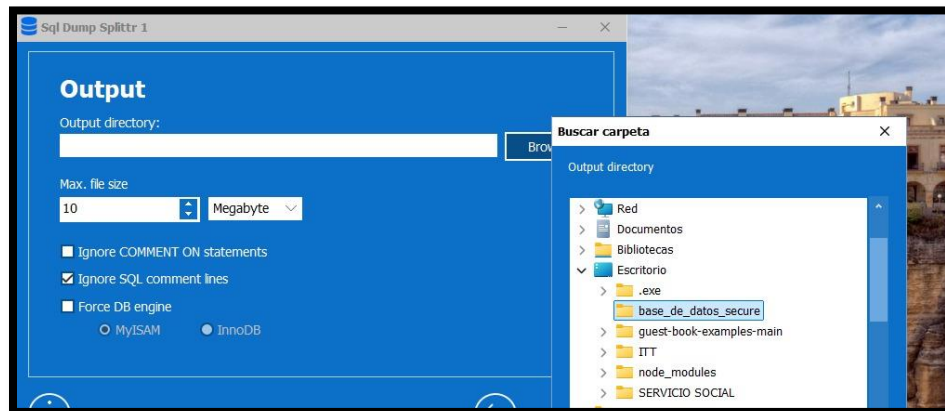


Ilustración 24: Ruta para guardar los archivos.

- Una vez completado con éxito el proceso de división de archivos en partes más pequeñas, continuamos con la importación de la base de datos en el nuevo servidor. Para ello, iniciamos XAMPP y creamos una nueva base de datos.

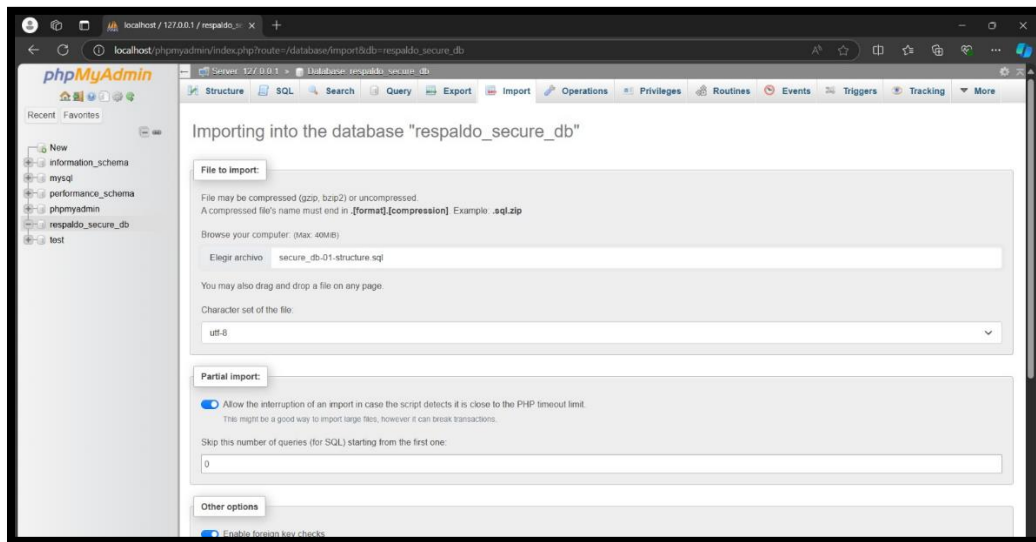


Ilustración 25: Base de datos nueva "respaldo_secure_db".

- Sobre la nueva base de datos, comenzamos a importar los archivos generados. En esta ocasión, se crearon 16 archivos, comenzando con el que se denomina 'estructura', seguido de los demás archivos enumerados del 1 al 16.

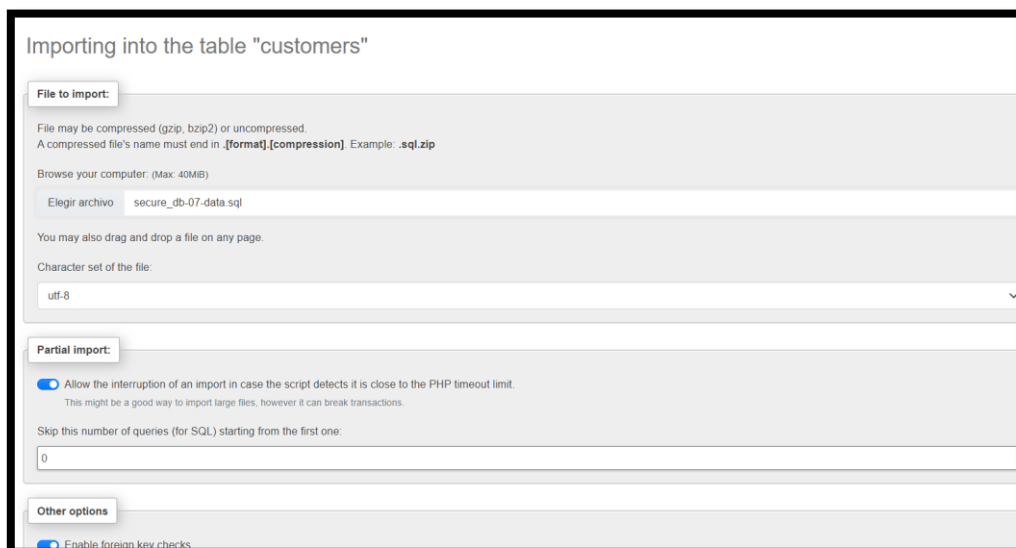
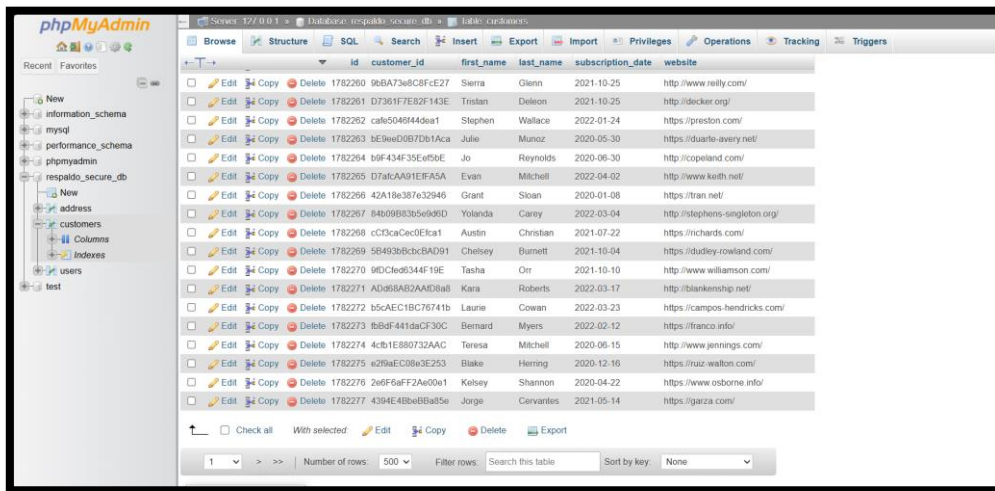


Ilustración 26: Importar archivos.

8. Verificamos que los registros se hayan insertado con éxito accediendo a la tabla 'customers', que es la que contiene más datos.

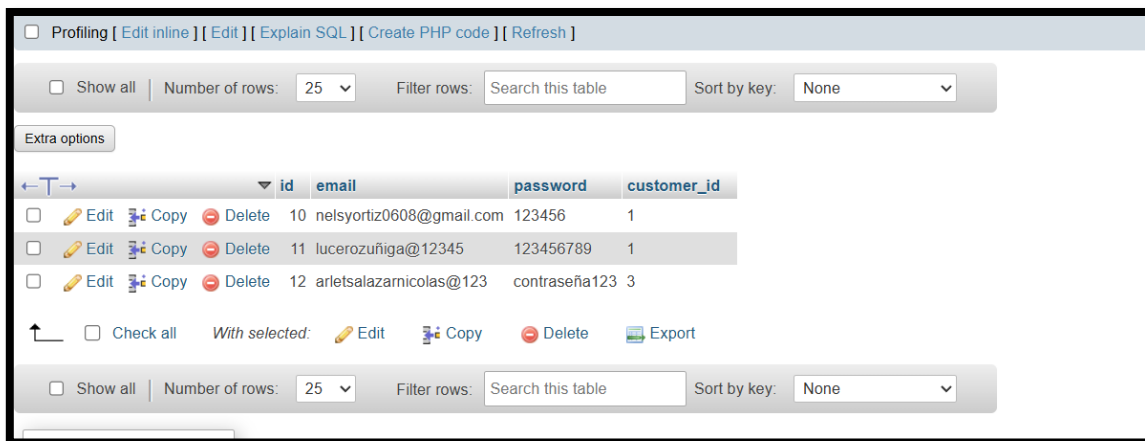


The screenshot shows the phpMyAdmin interface with the 'customers' table selected. The table has columns: id, customer_id, first_name, last_name, subscription_date, and website. The data is displayed in a table with 20 rows. Each row has action buttons (Edit, Copy, Delete) and a checkbox. The bottom of the interface shows 'Number of rows: 500' and a search bar.

	id	customer_id	first_name	last_name	subscription_date	website
<input type="checkbox"/>	1782260	96BA73e8C8F-cE27	Sierra	Glenn	2021-10-25	http://www.reilly.com/
<input type="checkbox"/>	1782261	D7361F7E82F-143E	Tristan	Deleon	2021-10-25	http://decker.org/
<input type="checkbox"/>	1782262	cafe504944dea1	Stephen	Wallace	2022-01-24	https://preston.com/
<input type="checkbox"/>	1782263	bE9eD087Dh1Aca	Julie	Munoz	2020-05-30	https://duarte-avery.net/
<input type="checkbox"/>	1782264	b8F434F35eE95E	Jo	Reynolds	2020-06-30	http://copeland.com/
<input type="checkbox"/>	1782265	D7aCA91EFa5A	Evan	Mitchell	2022-04-02	http://www.keith.net/
<input type="checkbox"/>	1782266	42A18e387e32946	Grant	Sloan	2020-01-08	https://tran.net/
<input type="checkbox"/>	1782267	84b09B83b5e9d0	Yolanda	Carey	2022-03-04	http://stephens-singleton.org/
<input type="checkbox"/>	1782268	cCF5caCec0Eka1	Austin	Christian	2021-07-22	https://richards.com/
<input type="checkbox"/>	1782269	5B493b8cbcbAD91	Chelsey	Burnett	2021-10-04	https://dudley-rowland.com/
<input type="checkbox"/>	1782270	9DC6ed9344F19E	Tasha	Orr	2021-10-10	http://www.willamson.com/
<input type="checkbox"/>	1782271	Abd98AB2AAD8a8	Kara	Roberts	2022-03-17	http://blankenship.net/
<input type="checkbox"/>	1782272	b5cAEC1BC76741b	Laune	Cowan	2022-03-23	https://campos-hendricks.com/
<input type="checkbox"/>	1782273	fb6d441daCF30C	Bernard	Myers	2022-02-12	https://franco.info/
<input type="checkbox"/>	1782274	4cb1E880732AAC	Teresa	Mitchell	2020-06-15	http://www.jennings.com/
<input type="checkbox"/>	1782275	e2f9aEC08a3E253	Blake	Herring	2020-12-16	https://ruiz-walton.com/
<input type="checkbox"/>	1782276	2e6F6aF2Ae00e1	Kelsey	Shannon	2020-04-22	https://www.osborne.info/
<input type="checkbox"/>	1782277	4394E4BbEBa85e	Jorge	Cervantes	2021-05-14	https://garza.com/

Ilustración 27: Datos de tabla Customers.

9. Verificamos que la tabla 'usuarios' contenga los datos.



The screenshot shows the phpMyAdmin interface with the 'usuarios' table selected. The table has columns: id, email, password, and customer_id. The data is displayed in a table with 3 rows. Each row has action buttons (Edit, Copy, Delete) and a checkbox. The bottom of the interface shows 'Number of rows: 25' and a search bar.

	id	email	password	customer_id
<input type="checkbox"/>	10	nelsyortiz0608@gmail.com	123456	1
<input type="checkbox"/>	11	lucerozuñiga@12345	123456789	1
<input type="checkbox"/>	12	arletsalazarnicolas@123	contraseña123	3

Ilustración 28: Datos de Tabla Users.

CONTENIDO

6. Investiga Y Describe Los Conceptos De Sql Injection Y Cómo Se Pueden Prevenir.	20
6.1 ¿Qué Es Un Ataque De Inyección?	20
6.2 Cómo Funciona El Sql Injection.....	20
6.3 Tipos De Sql Injection:.....	21
6.4 ¿Cómo Prevenir Un Ataque De Inyección Sql?	22
6.4.1 Uso De Consultas Parametrizadas (Prepared Statements):.....	22
6.4.2 Uso De Procedimientos Almacenados (Stored Procedures):	23
6.4.3. Validación De Entrada De Datos De Usuarios:	23
6.4.4 Medidas Adicionales:	23
7.1 Conceptos Claves De Bases De Datos Seguras:	24
Conclusión.	28
Referencias.	28

ÍNDICE DE ILUSTRACIONES

Ilustración 29: Ataques De Inyección.	20
Ilustración 30: Consecuencias De Un Ataque De Inyección.....	21
Ilustración 31: Prevenir Un Ataque De Inyección Sql.	22
Ilustración 32: Bases De Datos Seguras.....	24

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

6.1 ¿QUÉ ES UN ATAQUE DE INYECCION?



Ilustración 29: Ataques de Inyección.

SQL Injection es un tipo de ataque de seguridad web en el que un atacante inserta o "inyecta" código SQL malicioso en una consulta para acceder o manipular la base de datos subyacente de una aplicación. Este tipo de ataque explota vulnerabilidades en aplicaciones que no validan o filtran adecuadamente las entradas del usuario antes de utilizarlas en

consultas SQL.

El ataque se logra cuando una aplicación acepta datos de fuentes no confiables- que fueron alterados para ser interpretados como código- y además no realiza una correcta validación de los mismos antes de utilizarlos para realizar una consulta dinámica a la base de datos.

6.2 CÓMO FUNCIONA EL SQL INJECTION.

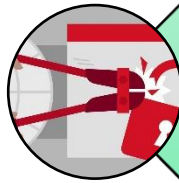
Cuando una aplicación construye una consulta SQL directamente a partir de la entrada proporcionada por el usuario, sin validación o escape adecuado, un atacante puede enviar comandos SQL maliciosos en lugar de los datos esperados. Esto puede resultar en:



Acceso no autorizado: Obtener acceso a datos sensibles, como credenciales de usuario o información financiera.



Manipulación de datos: Alterar o eliminar datos de la base de datos.



Escalación de privilegios: Ejecutar comandos de administrador en la base de datos.



Compromiso del sistema: En algunos casos, un atacante puede ejecutar comandos en el sistema operativo subyacente a través de la base de datos.

Ilustración 30: Consecuencias de un ataque de inyección.

6.3 TIPOS DE SQL INJECTION:

- **SQL Injection basado en errores:** El atacante provoca errores en la base de datos para obtener información.
- **SQL Injection ciego (Blind):** El atacante no recibe errores explícitos, pero puede inferir información a través de respuestas indirectas (tiempo de respuesta, cambios en la página, etc.).
- **Inyección de Unión (UNION-based):** Se usa para combinar el resultado de varias consultas y devolver información adicional de otras tablas.
- **Inyección por Tiempo (Time-based):** Los ataques se basan en ejecutar consultas SQL que retrasen la respuesta, permitiendo inferir si la inyección fue exitosa según los tiempos de respuesta.

6.4 ¿CÓMO PREVENIR UN ATAQUE DE INYECCIÓN SQL?

Existen distintos tipos de medidas que se deben considerar cuando se necesita comunicar un sistema con una base de datos relacional a través del uso de SQL.

- Uso de consultas parametrizadas: Prepared Statements.
- Uso de procedimientos: Stored Procedures.
- Validación de entrada de datos de usuarios
- Escapar todas las entradas permitidas de los usuarios.



Ilustración 31: Prevenir Un Ataque De Inyección SQL.

6.4.1 USO DE CONSULTAS PARAMETRIZADAS (PREPARED STATEMENTS):

Este es el enfoque más robusto para evitar inyecciones SQL, ya que permite que el motor de base de datos distinga claramente entre el código y los datos. La preparación previa de la consulta evita que las entradas del usuario puedan ser interpretadas como código ejecutable.

Los lenguajes más usados poseen métodos para parametrizar de manera segura las declaraciones que precisan datos de entrada de un usuario. Alguno de estos lenguajes es:

- Java EE – usar `PreparedStatement()` en nuestros parámetros dinámicos (variable Bind en inglés)
- .NET – usar consultas parametrizadas como `SqlCommand()` o `OleDbCommand()` en nuestros parámetros dinámicos.
- PHP – es posible usar PDO para base de datos genéricas con una fuerte parametrización de las consultas o en caso de usar un driver específico para una base de datos es necesario buscar una función segura para preparar nuestra declaración, por ejemplo, para MySQL es necesario usar `bind_param()`.

6.4.2 USO DE PROCEDIMIENTOS ALMACENADOS (STORED PROCEDURES):

Es importante que los procedimientos almacenados no generen consultas dinámicas que sean vulnerables. Si las entradas de los usuarios son pasadas directamente a consultas SQL dinámicas dentro del procedimiento, puede ocurrir una inyección SQL. Por tanto, es recomendable que los procedimientos almacenados solo utilicen entradas parametrizadas.

Control de acceso: Solo otorgar acceso a los procedimientos almacenados a las cuentas necesarias.

Revisión del código: Se deben revisar periódicamente los procedimientos almacenados para asegurar que no hay vulnerabilidades.

6.4.3. VALIDACIÓN DE ENTRADA DE DATOS DE USUARIOS:

Es fundamental validar las entradas de los usuarios antes de utilizarlas en cualquier consulta SQL, incluso si se están utilizando consultas parametrizadas. La validación debe incluir:

- **Tipos de datos:** Verificar que las entradas coincidan con el tipo de dato esperado.
- **Tamaños de entrada:** Limitar el tamaño de las entradas.
- **Listas blancas:** Para valores específicos como nombres de tablas o columnas, solo permitir aquellos que estén en una lista predeterminada.

6.4.4 MEDIDAS ADICIONALES:

- **Principio de menor privilegio:** Asegura que las cuentas de la base de datos tengan los permisos mínimos necesarios para la operación.
- **Uso de un firewall para aplicaciones web (WAF):** Un WAF puede ayudar a detectar y bloquear patrones de ataque comunes, incluyendo inyecciones SQL.

- **Monitoreo y registro de actividades (Logging):** Mantén un registro detallado de todas las consultas SQL ejecutadas y revisa regularmente los registros para identificar comportamientos sospechosos.

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

Una base de datos segura se refiere a un sistema de almacenamiento y gestión de datos diseñado con mecanismos para proteger la integridad, confidencialidad y disponibilidad de la información frente a ataques maliciosos, accesos no autorizados, pérdida o corrupción de datos. Implementar bases de datos seguras implica la adopción de prácticas, políticas y tecnologías que aseguren que los datos están protegidos en todo momento, tanto en reposo como en tránsito.

7.1 CONCEPTOS CLAVES DE BASES DE DATOS SEGURAS:



Ilustración 32: Bases de Datos Seguras.

Confidencialidad: Garantiza que los datos solo sean accesibles por personas autorizadas. Esto incluye encriptar datos sensibles y controlar el acceso a la base de datos.

Integridad: Asegura que los datos sean precisos y no sean alterados de manera no autorizada. La integridad también implica mecanismos que detecten y corrijan cualquier alteración no intencionada.

Disponibilidad: Garantiza que los datos estén disponibles cuando los usuarios autorizados los necesiten. Esto incluye planes de recuperación ante desastres y prevención contra ataques de denegación de servicio (DoS).

Autenticación: Se refiere a verificar la identidad de los usuarios que acceden a la base de datos para garantizar que solo personas legítimas puedan interactuar con ella.

Autorización: Determina qué operaciones puede realizar un usuario autenticado en la base de datos (leer, escribir, modificar, etc.).

Auditoría y monitoreo: Las bases de datos seguras incluyen mecanismos para registrar actividades y detectar comportamientos anómalos. Esto ayuda a identificar accesos no autorizados o potenciales ataques.

ESTRATEGIAS PARA IMPLEMENTAR BASES DE DATOS SEGURAS.

Autenticación: Implementar métodos robustos de autenticación, como autenticación multifactor (MFA), donde los usuarios deben proporcionar más de un tipo de credencial para acceder a la base de datos (contraseña, token, huella digital, etc.).

Autorización por roles: Aplicar políticas de control de acceso basado en roles (RBAC). Esto permite asignar diferentes permisos a los usuarios según sus funciones y responsabilidades dentro de la organización. Un administrador de bases de datos tendría más permisos que un usuario final, por ejemplo.

Encriptación de Datos:

Encriptación en reposo: Encriptar los datos cuando están almacenados para proteger la información sensible en caso de acceso no autorizado a los discos duros. La encriptación en reposo puede aplicarse tanto a los datos como a las copias de seguridad.

Encriptación en tránsito: Asegurar que los datos estén encriptados durante su transmisión a través de redes públicas o privadas mediante protocolos seguros como TLS (Transport Layer Security) o SSL (Secure Sockets Layer).

Encriptación a nivel de campo/columna: Encriptar datos específicos, como números de tarjetas de crédito o identificaciones personales, incluso dentro de la base de datos.

SEGURIDAD EN LA RED:

Firewalls: Implementar firewalls de base de datos y de red para controlar el tráfico que puede llegar a la base de datos y bloquear accesos no autorizados.

Segmentación de red: Separar la red donde se encuentra la base de datos de otras redes internas o externas para reducir la superficie de ataque. Los sistemas críticos deben estar en redes privadas y aisladas.

VPNs (Virtual Private Networks): Utilizar VPN para asegurar el acceso remoto a la base de datos, protegiendo los datos en tránsito entre la base de datos y el usuario.

PARCHEO Y ACTUALIZACIÓN:

Las bases de datos, al igual que otros sistemas, pueden ser vulnerables a fallos de seguridad si no se actualizan regularmente. Mantener actualizado el software de la base de datos y sus dependencias reduce la probabilidad de explotación de vulnerabilidades conocidas.

Auditoría y Monitoreo Continuo: Registro de actividades (logging): Registrar todas las operaciones importantes en la base de datos, como accesos, modificaciones y consultas. Esto ayuda a detectar posibles violaciones de seguridad o accesos no autorizados.

Monitoreo de actividades de bases de datos (DAM): Utilizar herramientas de monitoreo en tiempo real para detectar comportamientos sospechosos o no autorizados. Estas herramientas pueden alertar a los administradores de bases de datos de intentos de ataque, como inyecciones SQL o accesos fuera de lo común.

Copia de Seguridad Segura y Planes de Recuperación: Realizar copias de seguridad frecuentes de los datos y asegurarse de que estas estén almacenadas de manera segura (idealmente cifradas) en ubicaciones separadas. Implementar políticas de recuperación ante desastres para minimizar el tiempo de inactividad en caso de fallos.

Probar periódicamente las estrategias de recuperación para asegurarse de que los sistemas puedan ser restaurados eficazmente.

Separación de Datos y Funcionalidades: Evitar mezclar funciones de aplicaciones críticas con los servidores de bases de datos. Los servicios de base de datos deben ejecutarse en servidores dedicados para evitar que fallos en otras aplicaciones comprometan la seguridad de la base de datos.

Políticas de Contraseñas: Implementar políticas de contraseñas fuertes y seguras, como requerir combinaciones de caracteres especiales, números y letras mayúsculas y minúsculas, además de cambiar contraseñas periódicamente.

Principio de Menor Privilegio: Los usuarios solo deben tener los permisos mínimos necesarios para realizar sus tareas. No se debe otorgar acceso de administrador a usuarios que no lo necesiten, ni permisos excesivos que puedan poner en riesgo la seguridad del sistema.

Pruebas de Seguridad Regulares: Realizar auditorías de seguridad de manera continua para identificar y remediar vulnerabilidades antes de que sean explotadas. Esto incluye pruebas de penetración, análisis de vulnerabilidades y revisiones de configuración.

CONCLUSIÓN.

Las bases de datos seguras son esenciales para el desarrollo y gestión de sistemas. Los ingenieros en sistemas deben diseñar infraestructuras que protejan la integridad, confidencialidad y disponibilidad de la información. Técnicas como las consultas parametrizadas ayudan a prevenir ataques como las inyecciones SQL, mejorando la seguridad y eficiencia. También es crucial implementar controles de acceso y copias de seguridad para mantener la disponibilidad de los sistemas. Además, el uso de encriptación y firewalls protege la información de accesos no autorizados, garantizando así un entorno seguro. Por medio de la elaboración de esta práctica comprendimos que la seguridad en las bases de datos es fundamental para el funcionamiento seguro de cualquier sistema, especialmente en sectores donde la protección es esencial.

REFERENCIAS.

- *ibm.* (s.f.). Retrieved 15 de 09 de 2024, from <https://www.ibm.com/mx-es/topics/database-security>
- *kaspersky.* (s.f.). Retrieved 15 de 09 de 2024, from <https://latam.kaspersky.com/resource-center/definitions/sql-injection?srsId=AfmBOoolj6o-aeU0g8ZDY667yY73RGBR2ILKFdXNtlryaJmjchYWTgsw>
- LACNIC. (2020, 24 de marzo). *Cómo prevenir un ataque de inyección de SQL*. LACNIC.