Installation and Configuration of DVWA on Kali Linux VM

Introduction

This report outlines the complete step-by-step installation and configuration process of **DVWA (Damn Vulnerable Web Application)** on a **Kali Linux virtual machine**. DVWA is a deliberately vulnerable PHP/MySQL web application that is widely used by penetration testers, security analysts, and students to practice web application attacks, such as SQL Injection, XSS, CSRF, and Command Execution.

This setup was conducted manually in a secure and isolated **Kali Linux virtual machine** environment to ensure no risk to the host operating system or any external systems.

Disclaimer

DVWA is intentionally designed with multiple security flaws. It must **never be installed on a live production system or exposed to the public internet**, as it can be exploited by attackers and malware. Even within a virtual environment, if the system is misconfigured or if payloads are improperly handled, there is still a risk of infecting the host system or other devices on the network.

This installation should be performed **only for educational and ethical hacking purposes** in a safe lab environment. Misuse of DVWA or any related tools for unauthorized hacking is **illegal** and punishable under cybersecurity laws.

Prerequisite Knowledge

Before setting up DVWA, it is important to have a foundational understanding of **SQL**. This includes knowing what a **database**, **table**, **user**, and **privileges** are, and how PHP interacts with MySQL through configuration files. These concepts are critical for correctly creating and connecting the backend database that DVWA depends on.

Environment Preparation

The DVWA setup was initiated by first creating a **Kali Linux virtual machine** using a virtualization platform such as **VirtualBox** or **UTM**. Using a virtual machine is essential for creating an isolated environment where vulnerable applications can be safely executed and tested without compromising the host system.

After launching the VM, the system was updated using the command:

sudo apt update && sudo apt upgrade -y

This command ensures that all package indexes and dependencies are current, which helps avoid version conflicts or missing dependencies later in the installation process.

Database Installation: MySQL and MariaDB

The next step was installing a database management system. In this setup, **MySQL Server** was installed using:

sudo apt install mysql-server -y

MySQL is a relational database management system that stores data in structured tables. DVWA uses MySQL to store user data, captured inputs, and logs of vulnerabilities.

It is important to note that on many Linux distributions, including Kali, **MariaDB** may be used instead of MySQL. MariaDB is a **drop-in replacement** for MySQL — it is an open-source, community-developed fork of MySQL that maintains compatibility with MySQL commands, libraries, and interfaces. Many systems ship with MariaDB by default due to its open licensing, better performance improvements, and ongoing community support. Whether MySQL or MariaDB is installed, the configuration for DVWA remains largely the same.

Web Server and Directory Setup

Once MySQL was installed, the root user session was initiated using:

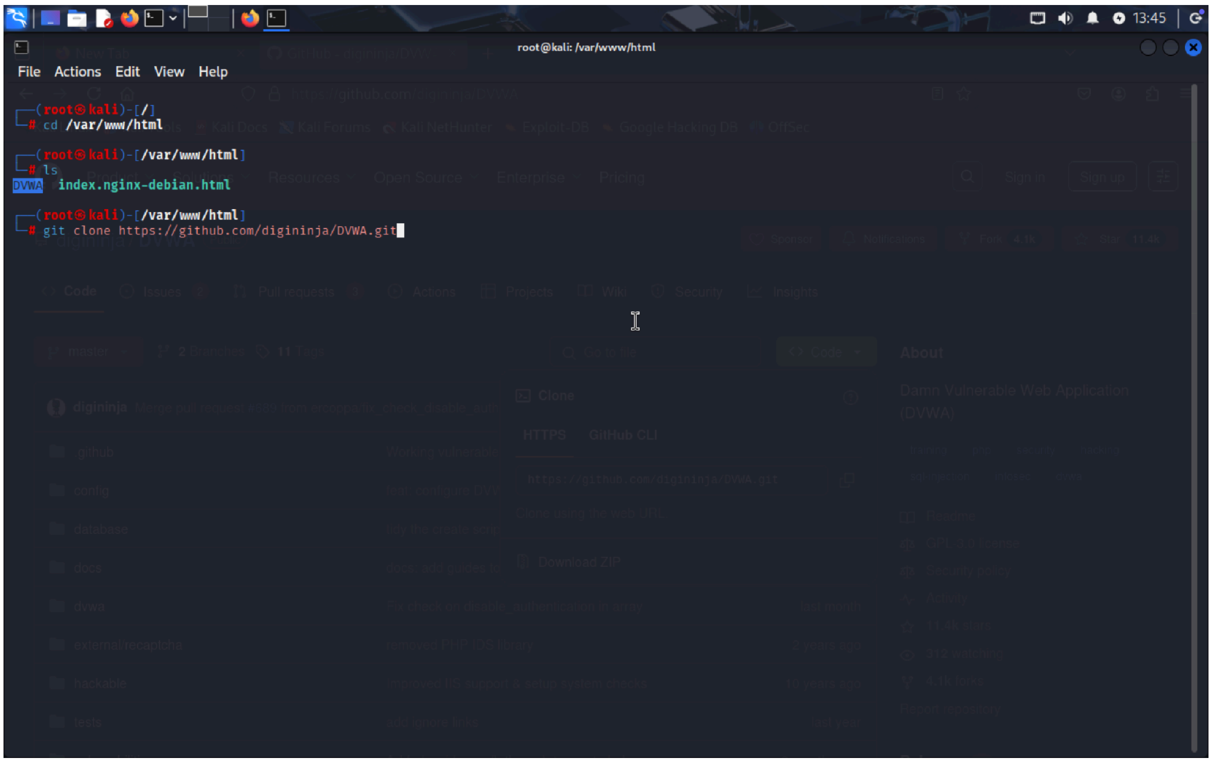
sudo su

This command switches the terminal to superuser mode, granting full administrative privileges required for modifying the web server directory and configuration files.

The web server root directory is located at:

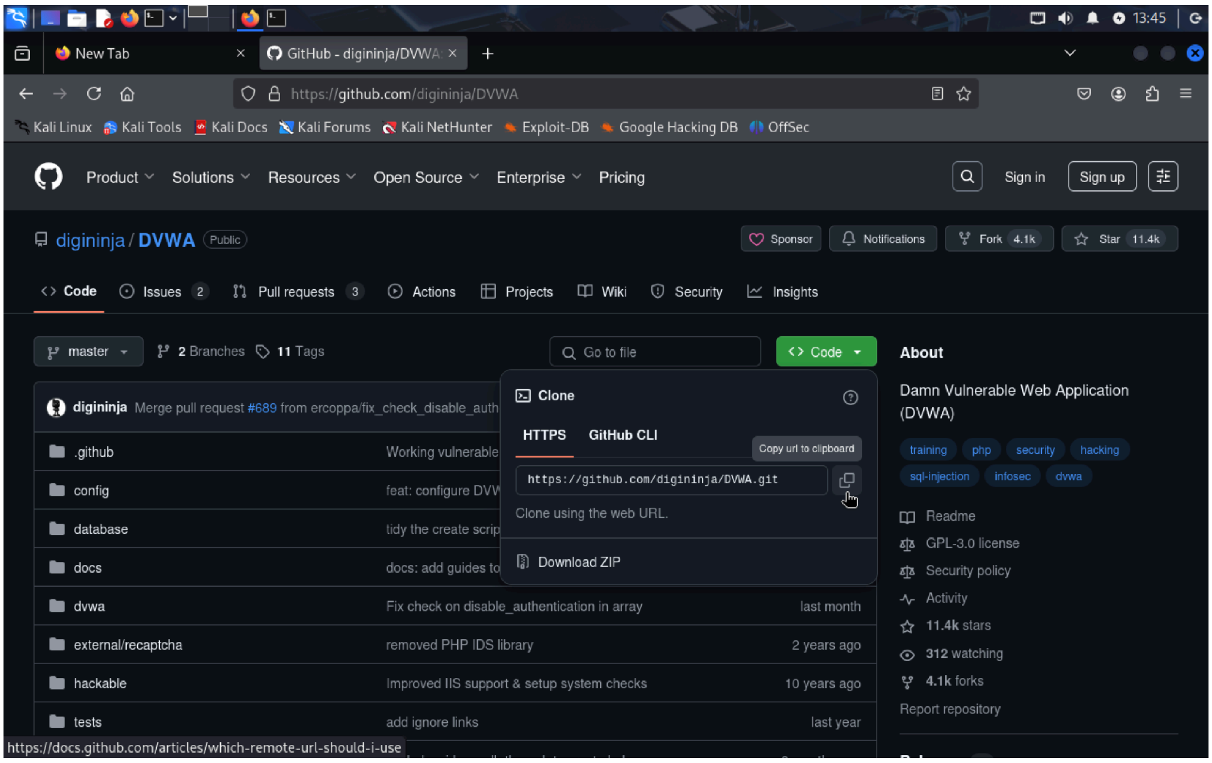
/var/www/html

Navigating to this directory is necessary because it is where **Apache** serves web applications. Inside this directory, DVWA was downloaded using:



```
┌──(root㉿kali)-[/]
└─# cd /var/www/html

┌──(root㉿kali)-[/var/www/html]
└─# ls
DVWA  index.nginx-debian.html

┌──(root㉿kali)-[/var/www/html]
└─# git clone https://github.com/digininja/DVWA.git
```

Fig: Cloning DVWA



digininja Merge pull request #689 from ercoppa/fix_check_disable_auth

.github	Working vulnerable	
config	feat: configure DVW	
database	tidy the create scrip	
docs	docs: add guides to	
dvwa	Fix check on disable_authentication in array	last month
external/recaptcha	removed PHP IDS library	2 years ago
hackable	Improved IIS support & setup system checks	10 years ago
tests	add ignore links	last year

```
https://github.com/digininja/DVWA.git
```

Clone using the web URL.

Damn Vulnerable Web Application (DVWA)

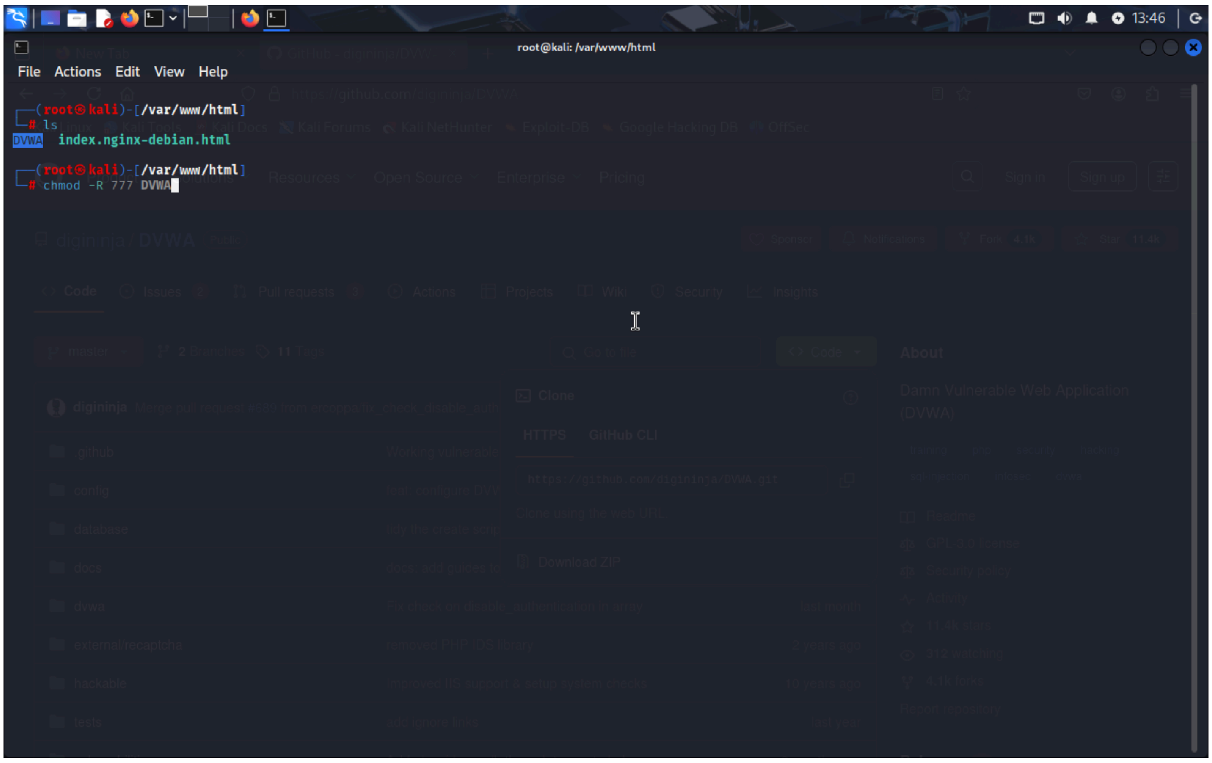
https://docs.github.com/articles/which-remote-url-should-i-use

Fig: DVWA URL LINK

git clone https://github.com/digininja/DVWA.git

This command clones the official DVWA source code from GitHub into the Apache web root.

To avoid permission errors during testing, full read-write-execute permissions were granted using:



```
┌──(root㉿kali)-[/var/www/html]
└─# ls
DVWA  index.nginx-debian.html

┌──(root㉿kali)-[/var/www/html]
└─# chmod -R 777 DVWA
```

Fig: Grant Permissions

chmod -R 777 DVWA

Although 777 permissions are insecure and never recommended for production, they simplify the testing process in isolated environments by eliminating permission restrictions.

DVWA Configuration

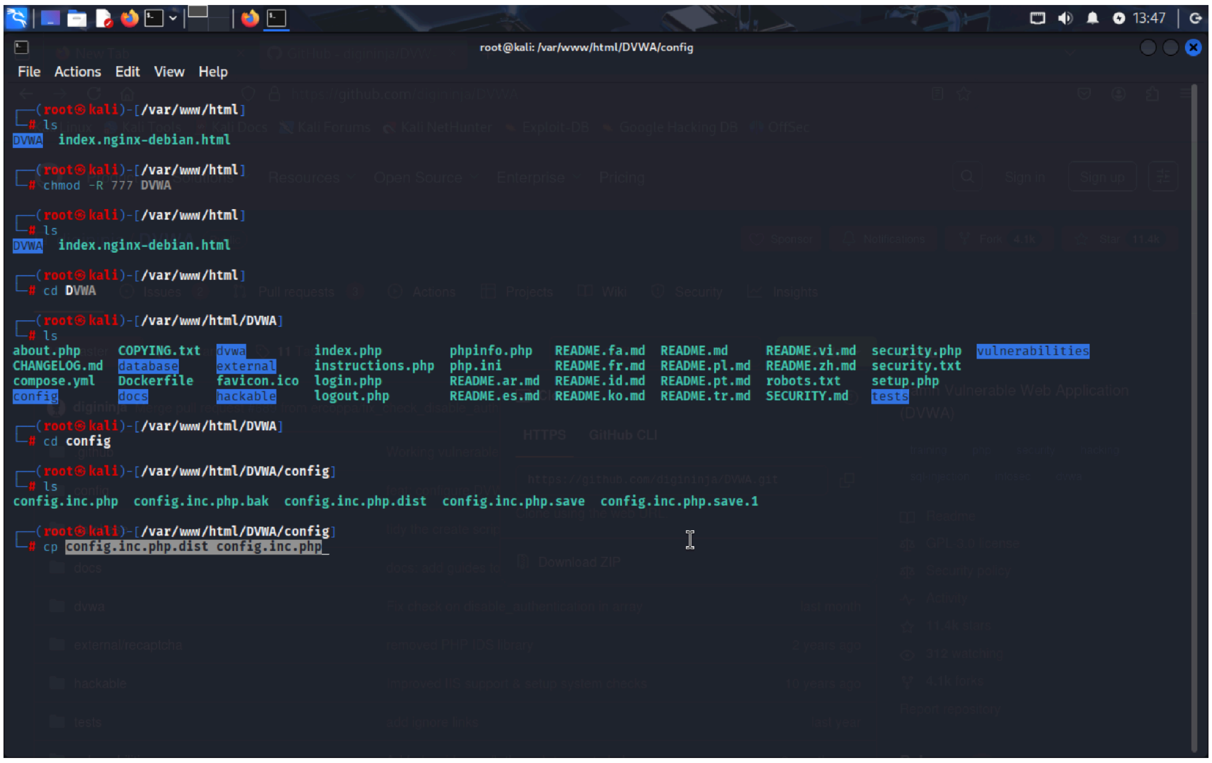
Inside the DVWA/config folder, the sample configuration file named config.inc.php.dist was copied and renamed to config.inc.php. This is the primary configuration file where database connection details are defined.

The file was edited using the command:

nano config.inc.php

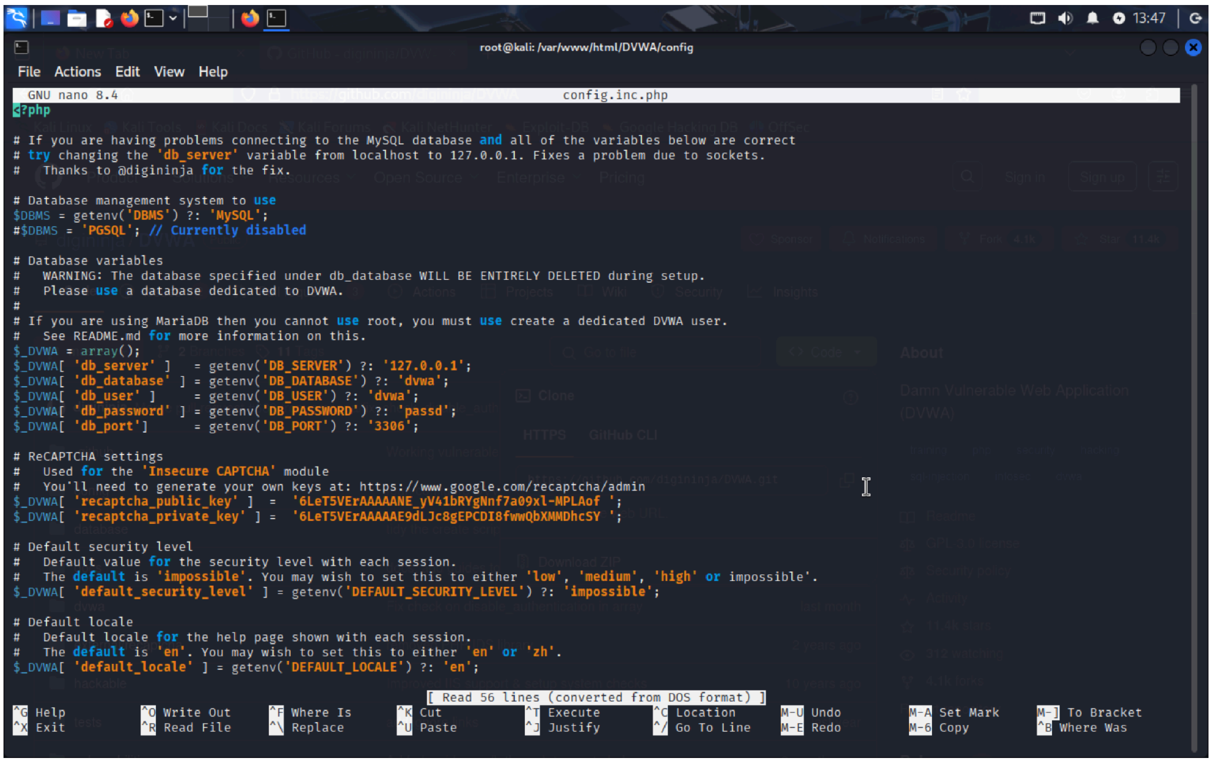
The following parameters were set:

$_DVWA['db_user'] = 'dvwa';



```
root@kali: /var/www/html/DVWA/config

┌──(root㉿kali)-[/var/www/html]
└─# ls
DVWA  index.nginx-debian.html

┌──(root㉿kali)-[/var/www/html]
└─# chmod -R 777 DVWA

┌──(root㉿kali)-[/var/www/html]
└─# ls
DVWA  index.nginx-debian.html

┌──(root㉿kali)-[/var/www/html]
└─# cd DVWA

┌──(root㉿kali)-[/var/www/html/DVWA]
└─# ls
about.php     COPYING.txt  dvwa         index.php         phpinfo.php   README.fa.md  README.md     README.vi.md  security.php  vulnerabilities
CHANGELOG.md  database     external     instructions.php  php.ini       README.fr.md  README.pl.md  README.zh.md  security.txt
compose.yml   Dockerfile   favicon.ico  login.php         README.ar.md  README.id.md  README.pt.md  robots.txt    setup.php
config        docs         hackable     logout.php        README.es.md  README.ko.md  README.tr.md  SECURITY.md   tests

┌──(root㉿kali)-[/var/www/html/DVWA]
└─# cd config

┌──(root㉿kali)-[/var/www/html/DVWA/config]
└─# ls
config.inc.php  config.inc.php.bak  config.inc.php.dist  config.inc.php.save  config.inc.php.save.1

┌──(root㉿kali)-[/var/www/html/DVWA/config]
└─# cp config.inc.php.dist config.inc.php
```

Fig: copy the distribution template file



```
root@kali: /var/www/html/DVWA/config

  GNU nano 8.4                                   config.inc.php
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
#   Thanks to @digininja for the fix.

# Database management system to use
$DBMS = getenv('DBMS') ?: 'MySQL';
#$DBMS = 'PGSQL'; // Currently disabled

# Database variables
#   WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
#   Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
#   See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ]   = getenv('DB_SERVER') ?: '127.0.0.1';
$_DVWA[ 'db_database' ] = getenv('DB_DATABASE') ?: 'dvwa';
$_DVWA[ 'db_user' ]     = getenv('DB_USER') ?: 'dvwa';
$_DVWA[ 'db_password' ] = getenv('DB_PASSWORD') ?: 'passd';
$_DVWA[ 'db_port']      = getenv('DB_PORT') ?: '3306';

# ReCAPTCHA settings
#   Used for the 'Insecure CAPTCHA' module
#   You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA[ 'recaptcha_public_key' ]  =  '6LeT5VErAAAAANE_yV41bRYgNnf7a09xl-MPLAof ';
$_DVWA[ 'recaptcha_private_key' ] =  '6LeT5VErAAAAAE9dLJc8gEPCDI8fwwQbXMMDhcSY ';

# Default security level
#   Default value for the security level with each session.
#   The default is 'impossible'. You may wish to set this to either 'low', 'medium', 'high' or impossible'.
$_DVWA[ 'default_security_level' ] = getenv('DEFAULT_SECURITY_LEVEL') ?: 'impossible';

# Default locale
#   Default locale for the help page shown with each session.
#   The default is 'en'. You may wish to set this to either 'en' or 'zh'.
$_DVWA[ 'default_locale' ] = getenv('DEFAULT_LOCALE') ?: 'en';

                          [ Read 56 lines (converted from DOS format) ]
^G Help       ^O Write Out  ^F Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^B Where Was
```

Fig: DVWA config file

$_DVWA['db_password'] = 'dvwapass';

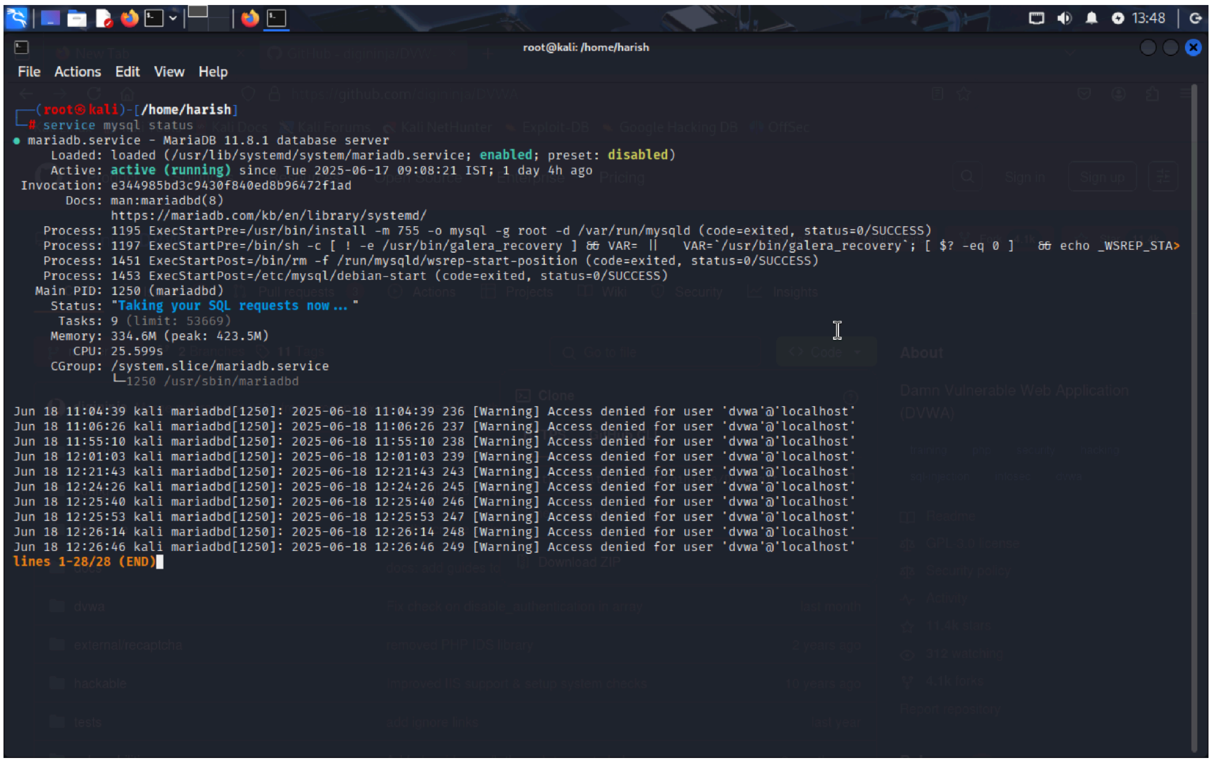
$_DVWA['db_database'] = 'dvwa';

These credentials must match exactly with the database and user that will be created in MySQL. If there is any mismatch, DVWA will fail to connect to the database and return an error.

Database Creation and User Management

The MySQL service was started using:

service mysql start

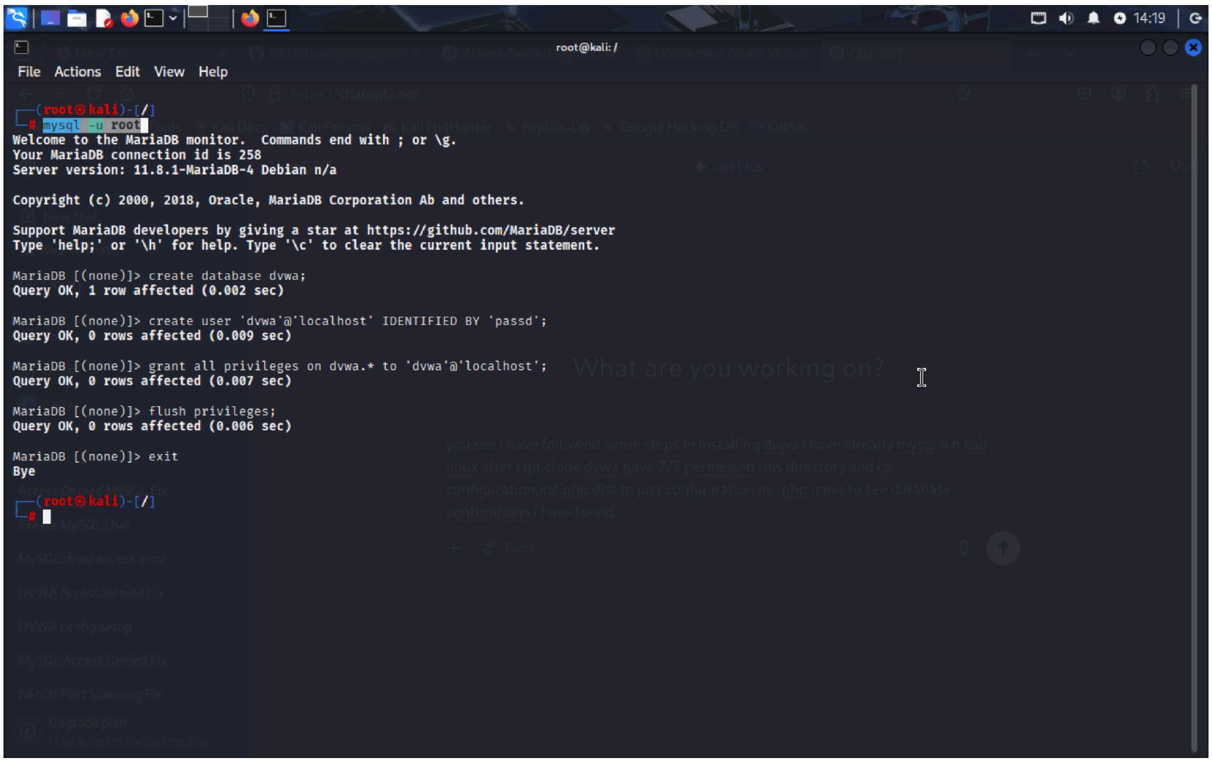


```
┌──(root㉿kali)-[/home/harish]
└─# service mysql status
● mariadb.service - MariaDB 11.8.1 database server
     Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: disabled)
     Active: active (running) since Tue 2025-06-17 09:08:21 IST; 1 day 4h ago
 Invocation: e344985bd3c9430f840ed8b96472f1ad
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
    Process: 1195 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysqld (code=exited, status=0/SUCCESS)
    Process: 1197 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR= ||   VAR=`/usr/bin/galera_recovery`; [ $? -eq 0 ]   && echo _WSREP_STA>
    Process: 1451 ExecStartPost=/bin/rm -f /run/mysqld/wsrep-start-position (code=exited, status=0/SUCCESS)
    Process: 1453 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
   Main PID: 1250 (mariadbd)
     Status: "Taking your SQL requests now... "
      Tasks: 9 (limit: 53669)
     Memory: 334.6M (peak: 423.5M)
        CPU: 25.599s
     CGroup: /system.slice/mariadb.service
             └─1250 /usr/sbin/mariadbd

Jun 18 11:04:39 kali mariadbd[1250]: 2025-06-18 11:04:39 236 [Warning] Access denied for user 'dvwa'@'localhost'
Jun 18 11:06:26 kali mariadbd[1250]: 2025-06-18 11:06:26 237 [Warning] Access denied for user 'dvwa'@'localhost'
Jun 18 11:55:10 kali mariadbd[1250]: 2025-06-18 11:55:10 238 [Warning] Access denied for user 'dvwa'@'localhost'
Jun 18 12:01:03 kali mariadbd[1250]: 2025-06-18 12:01:03 239 [Warning] Access denied for user 'dvwa'@'localhost'
Jun 18 12:21:43 kali mariadbd[1250]: 2025-06-18 12:21:43 243 [Warning] Access denied for user 'dvwa'@'localhost'
Jun 18 12:24:26 kali mariadbd[1250]: 2025-06-18 12:24:26 245 [Warning] Access denied for user 'dvwa'@'localhost'
Jun 18 12:25:40 kali mariadbd[1250]: 2025-06-18 12:25:40 246 [Warning] Access denied for user 'dvwa'@'localhost'
Jun 18 12:25:53 kali mariadbd[1250]: 2025-06-18 12:25:53 247 [Warning] Access denied for user 'dvwa'@'localhost'
Jun 18 12:26:14 kali mariadbd[1250]: 2025-06-18 12:26:14 248 [Warning] Access denied for user 'dvwa'@'localhost'
Jun 18 12:26:46 kali mariadbd[1250]: 2025-06-18 12:26:46 249 [Warning] Access denied for user 'dvwa'@'localhost'
lines 1-28/28 (END)
```

Fig: Mysql Service

Then, the MySQL console was accessed using:

mysql -u root



```
┌──(root㉿kali)-[/]
└─# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 258
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 dvwa;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> create user 'dvwa'@'localhost' IDENTIFIED BY 'passd';
Query OK, 0 rows affected (0.009 sec)

MariaDB [(none)]> grant all privileges on dvwa.* to 'dvwa'@'localhost';
Query OK, 0 rows affected (0.007 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.006 sec)

MariaDB [(none)]> exit
Bye

┌──(root㉿kali)-[/]
└─# 
```

Fig: Create DATABASE

Inside the MySQL prompt, the DVWA database and user were created using the following commands:

CREATE DATABASE dvwa;

CREATE USER 'dvwa'@'localhost' IDENTIFIED BY 'dvwapass';

GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwa'@'localhost';

FLUSH PRIVILEGES;

These commands create a new database named dvwa, a user named dvwa, and assign full privileges on the database to that user. The FLUSH PRIVILEGES command reloads the grant tables to apply changes.

The database user must match the one declared in config.inc.php. Any mismatch between the MySQL user and the configuration file will result in a **database connection error** when launching DVWA.

Starting Web Server and PHP Configuration

The Apache web server was started using:

service apache2 start

Alternatively, services can be managed more effectively using the systemctl command. To automatically start services on system boot, the following command is used:

systemctl enable apache2

Enabling services with systemctl ensures they are persistently started during every system startup. Without this, the service must be manually started each time the system boots.

PHP is the server-side language that processes DVWA's logic. To support remote file inclusion and other vulnerable features, the PHP configuration file was modified. The file is located at:

/etc/php/<version>/apache2/php.ini

Within the file, the following directives were set:

allow_url_fopen = On

allow_url_include = On

These directives allow PHP to fetch and include remote files, which is necessary for simulating vulnerabilities such as Remote File Inclusion (RFI). After changes were made, Apache was restarted using:

service apache2 restart

Launching DVWA in Browser

Once all services were running, DVWA was accessed in a browser via the URL:

http://localhost/DVWA/setup.php

This page contains the setup wizard to initialize the DVWA database structure. Clicking the **"Create / Reset Database"** button populates the tables and prepares the application for use.

The login screen was then accessed at:

http://localhost/DVWA/login.php

The default credentials are:

Username: admin

Password: password

Successful login indicates that DVWA has been fully installed and is ready for testing.

Common Configuration Errors

During setup, one frequent issue encountered is that Apache may fail to start and display an error related to **HTTP port 80**. This usually happens if another service is already occupying port 80, or if Apache is already running in the background.

The error message may resemble:

```
AH00558: apache2: Could not reliably determine the server's fully qualified domain name
```

or

```
(98)Address already in use: AH00072: make_sock: could not bind to address [::]:80
```

To resolve this, it is necessary to check if another process is using port 80 by running:

```
sudo lsof -i :80
```

If Apache is already running, a simple restart can resolve it:

sudo systemctl restart apache2

If another service is occupying port 80, that service must be stopped or Apache must be reconfigured to use a different port.

Conclusion

This report documented the end-to-end installation of DVWA on Kali Linux using manual configuration methods. It covered all steps from system update, database installation, application cloning, permission setting, database user creation, and PHP configuration, to launching the application in a browser. Emphasis was placed on why each step is necessary, what configurations affect DVWA's functionality, and how to troubleshoot common errors.

This lab setup provides a robust environment for understanding web application vulnerabilities and sharpening ethical hacking skills. Proper configuration and environment isolation are critical to ensure that vulnerable applications like DVWA are used safely and legally.