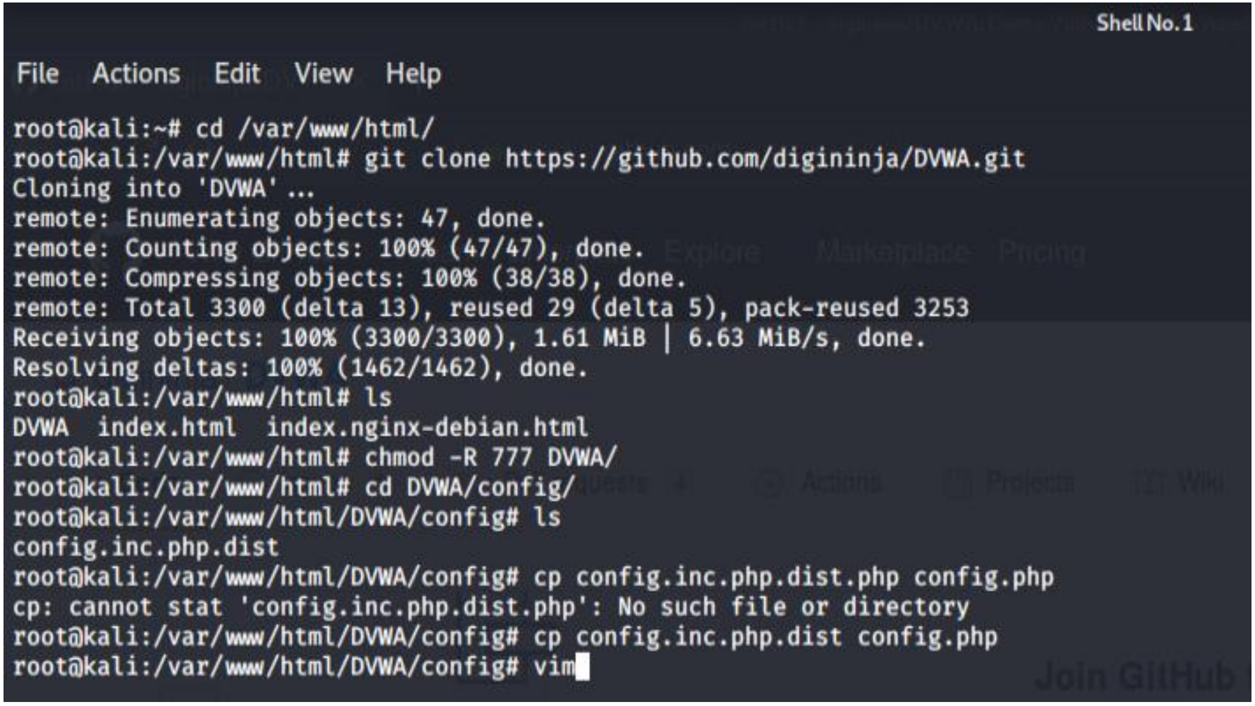
Cross Site Scripting Lab

1. **Overview**
   The main objectives of this lab are Cross-Site Scripting Attacks (XSS). The XSSs are a type of injection attack, client-side injection attack, where malicious scripts are injected into trusted websites.

2. **Resources Required**
   The first requirement of this system is Damn Vulnerable Web Application (DVWA) and XAMPP packages. DVWA is open source open application designed to be vulnerable that runs on local server. I followed the YouTube link provided in the lab instruction and installed DVWA and XAMPP packages in my Linux machine.

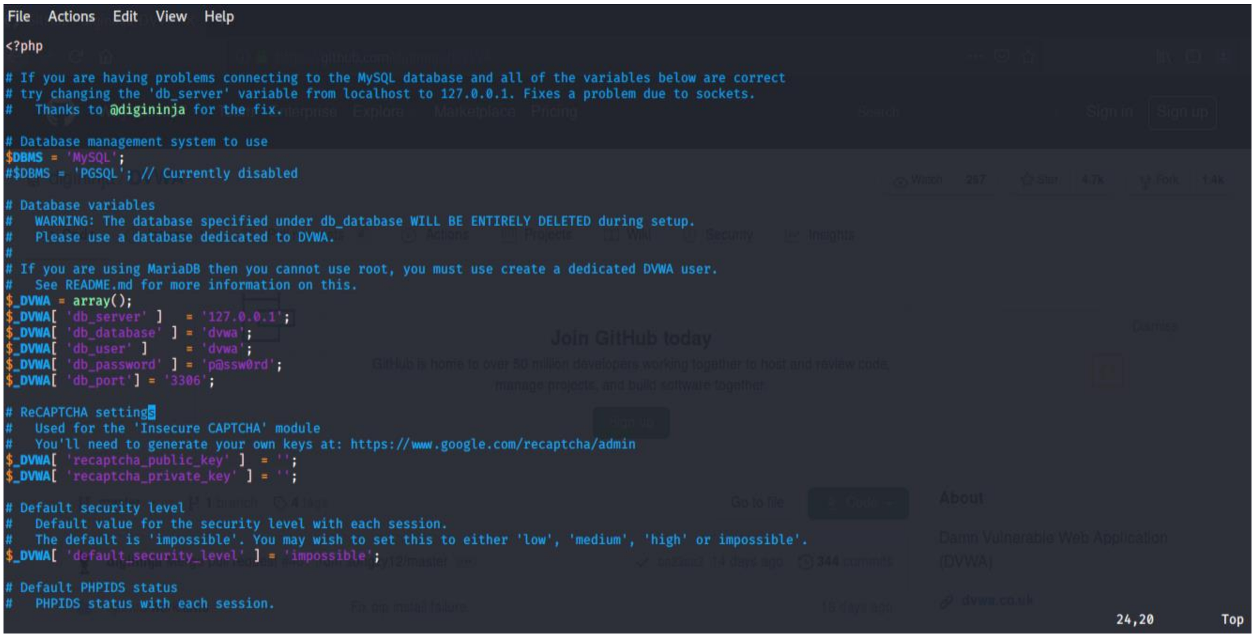
3. **Initial Setup**
   First, I cloned a github Repo into my Kali machine under /var/www/html/ because all files should be there to run in the local server. I gave all permission to all the files in the directory DVWA.



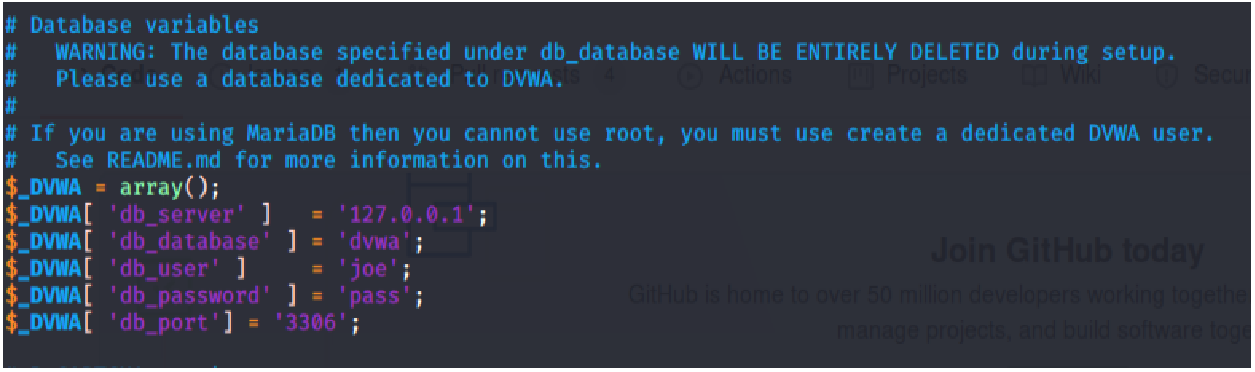
```
root@kali:~# cd /var/www/html/
root@kali:/var/www/html# git clone https://github.com/digininja/DVWA.git
Cloning into 'DVWA'...
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 3300 (delta 13), reused 29 (delta 5), pack-reused 3253
Receiving objects: 100% (3300/3300), 1.61 MiB | 6.63 MiB/s, done.
Resolving deltas: 100% (1462/1462), done.
root@kali:/var/www/html# ls
DVWA  index.html  index.nginx-debian.html
root@kali:/var/www/html# chmod -R 777 DVWA/
root@kali:/var/www/html# cd DVWA/config/
root@kali:/var/www/html/DVWA/config# ls
config.inc.php.dist
root@kali:/var/www/html/DVWA/config# cp config.inc.php.dist.php config.php
cp: cannot stat 'config.inc.php.dist.php': No such file or directory
root@kali:/var/www/html/DVWA/config# cp config.inc.php.dist config.php
root@kali:/var/www/html/DVWA/config# vim
```

Then, I located the configuration php file and made a copy of it. The php file looked like this initially. I changed user to ‘joe’ and password to ‘pass’ using vim editor as follows.

Cross-Site Scripting Lab

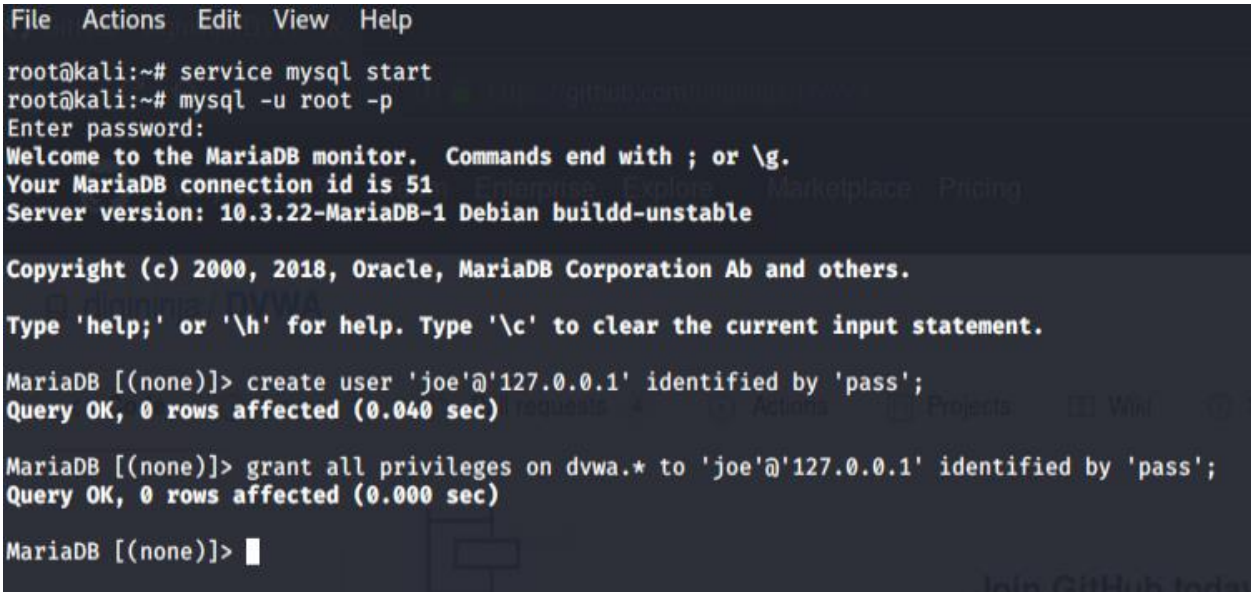


```
<?php

# 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
$DBMS = 'MySQL';
#$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' ]   = '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ]     = 'dvwa';
$_DVWA[ 'db_password' ] = 'p@ssw0rd';
$_DVWA[ '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' ]  = '';
$_DVWA[ 'recaptcha_private_key' ] = '';

# 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' ] = 'impossible';

# Default PHPIDS status
#   PHPIDS status with each session.
                                                                              24,20         Top
```



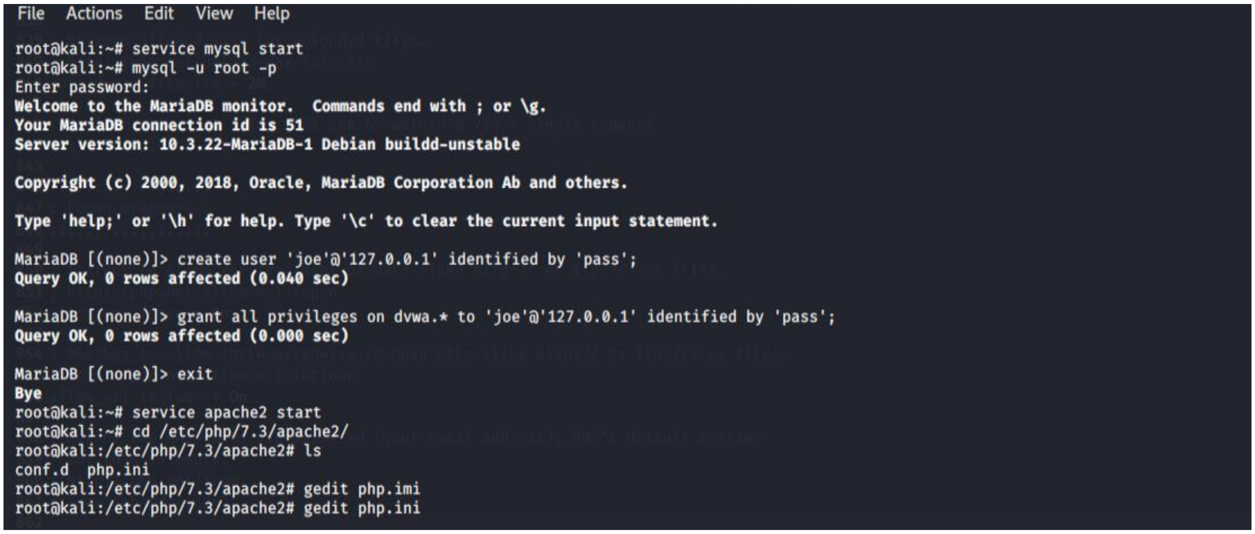
```
# Database variables
#   WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
#   Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
#   See README.md for more information on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

I saved the changes and exited. Then I created and configured a database using MySQL as follows.



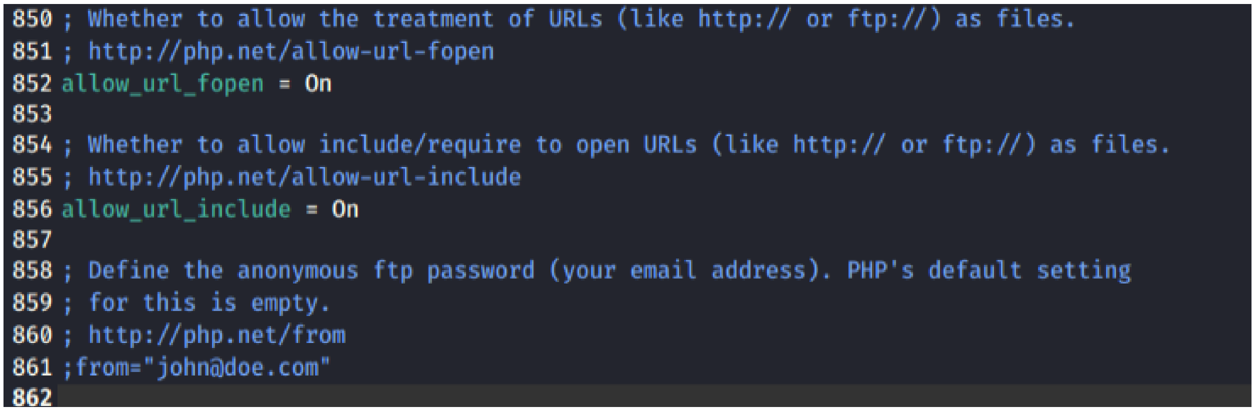
```
root@kali:~# service mysql start
root@kali:~# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 51
Server version: 10.3.22-MariaDB-1 Debian buildd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create user 'joe'@'127.0.0.1' identified by 'pass';
Query OK, 0 rows affected (0.040 sec)

MariaDB [(none)]> grant all privileges on dvwa.* to 'joe'@'127.0.0.1' identified by 'pass';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> 
```

Cross-Site Scripting Lab



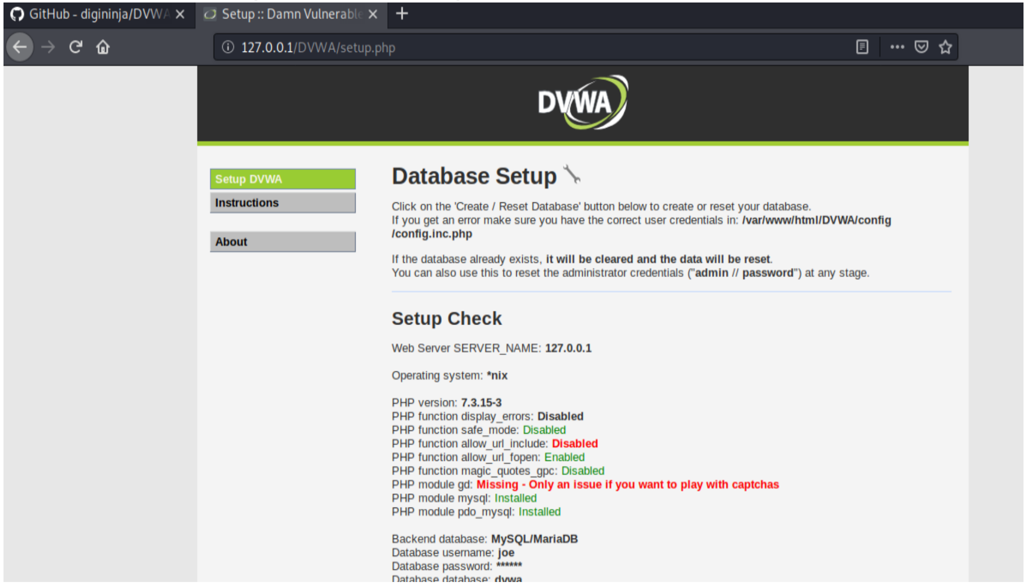
```
root@kali:~# service mysql start
root@kali:~# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 51
Server version: 10.3.22-MariaDB-1 Debian buildd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create user 'joe'@'127.0.0.1' identified by 'pass';
Query OK, 0 rows affected (0.040 sec)

MariaDB [(none)]> grant all privileges on dvwa.* to 'joe'@'127.0.0.1' identified by 'pass';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> exit
Bye
root@kali:~# service apache2 start
root@kali:~# cd /etc/php/7.3/apache2/
root@kali:/etc/php/7.3/apache2# ls
conf.d  php.ini
root@kali:/etc/php/7.3/apache2# gedit php.imi
root@kali:/etc/php/7.3/apache2# gedit php.ini
```

Once the SQL is up and running, I configured apache server by editing the allow url to ‘On’ as follows.



```
; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-fopen
allow_url_fopen = On

; Whether to allow include/require to open URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-include
allow_url_include = On

; Define the anonymous ftp password (your email address). PHP's default setting
; for this is empty.
; http://php.net/from
;from="john@doe.com"

```

After the configuration is done for apache server, I typed “**127.0.0.1/DVWA/**” on the web browser and I got the setup page as follows.



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/DVWA/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: **127.0.0.1**

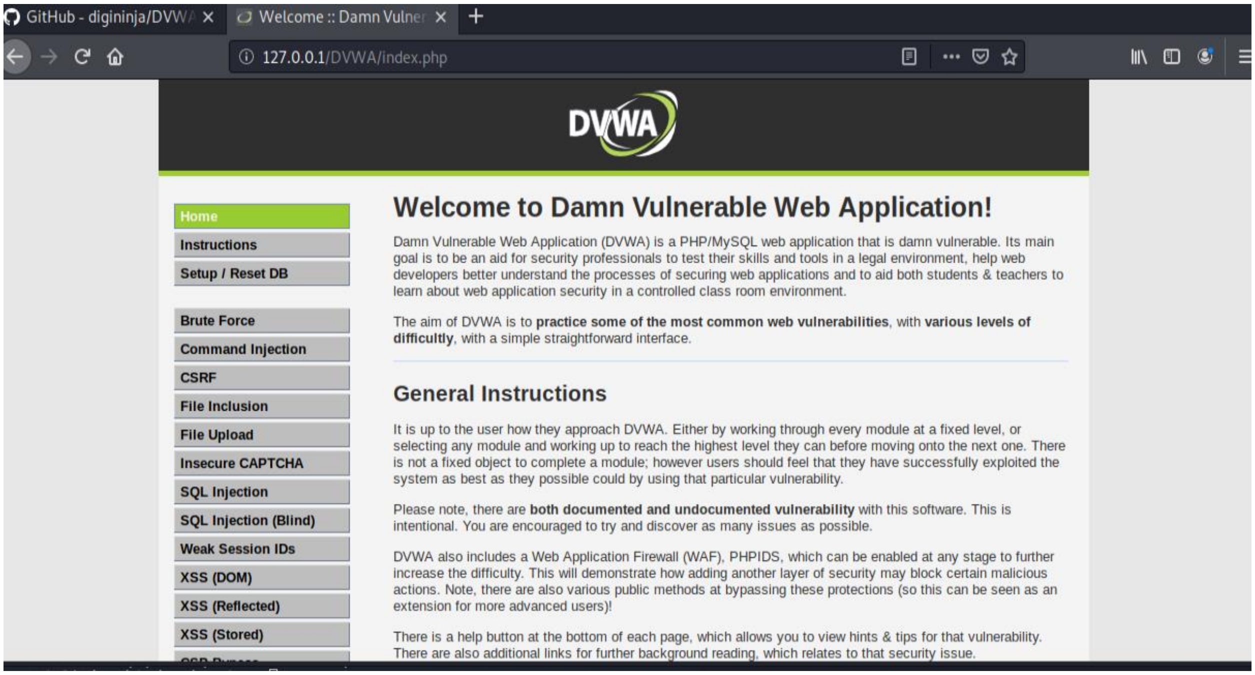
Operating system: ***nix**

PHP version: **7.3.15-3**
PHP function display_errors: **Disabled**
PHP function safe_mode: Disabled
PHP function allow_url_include: **Disabled**
PHP function allow_url_fopen: Enabled
PHP function magic_quotes_gpc: Disabled
PHP module gd: **Missing - Only an issue if you want to play with captchas**
PHP module mysql: Installed
PHP module pdo_mysql: Installed

Backend database: **MySQL/MariaDB**
Database username: **joe**
Database password: **********
Database database: **dvwa**

Cross-Site Scripting Lab

I did initial setup and logged in with username= “admin” and password= “password”. Finally, I was inside the DVWA web application on our local host running on the apache2 server.



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

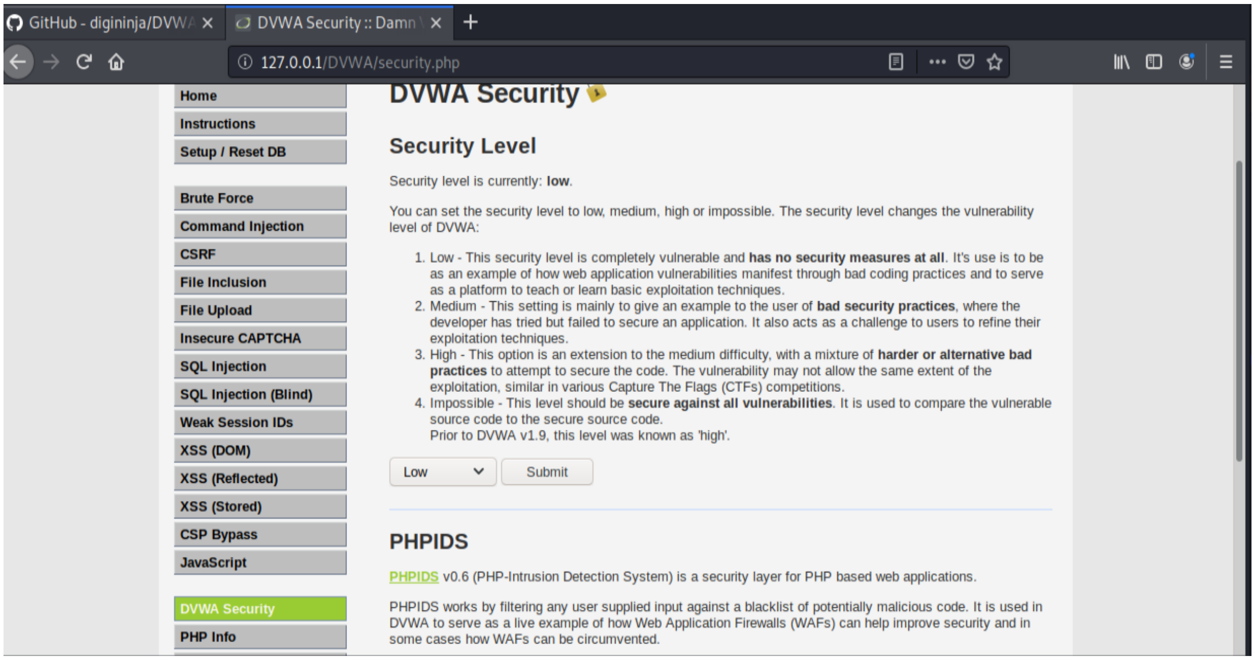
It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users)!

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

I set the security level to “low’ as follows:



DVWA Security

Security Level

Security level is currently: **low**.

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

PHPIDS

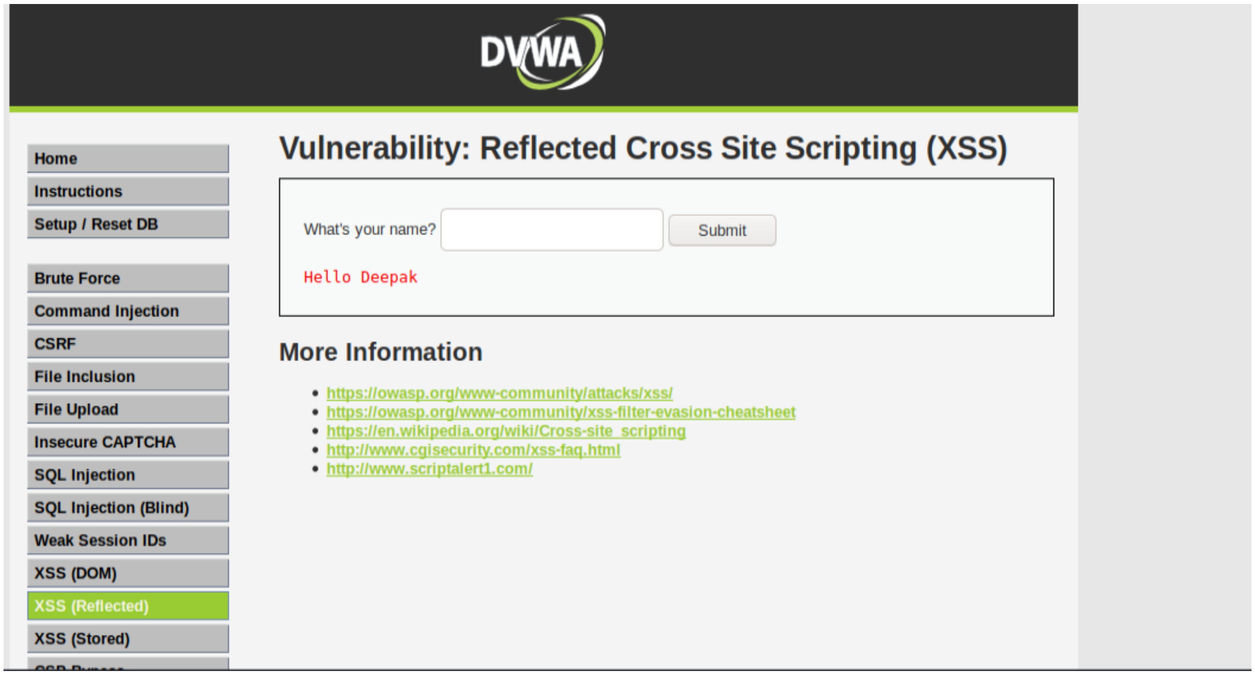
PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

Cross-Site Scripting Lab

4. Tasks

For our cross-scripting task, I went to the "XSS(Reflected" tab and tried typing my name and my name was reflected as follows.



Vulnerability: Reflected Cross Site Scripting (XSS)

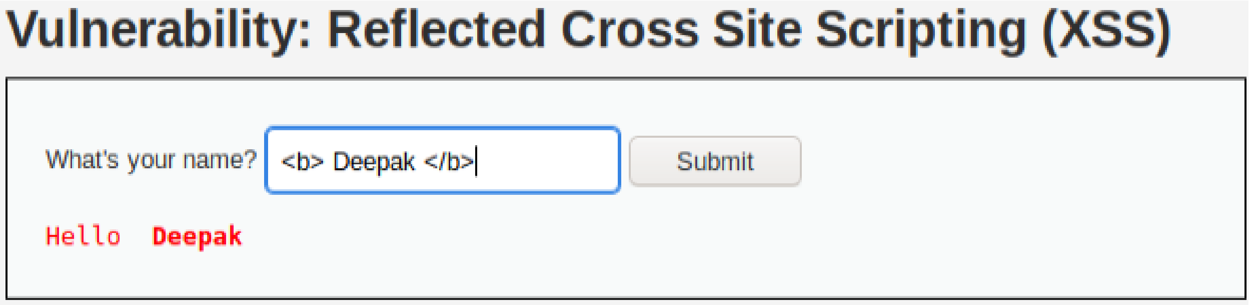
What's your name?

Hello Deepak

More Information

- https://owasp.org/www-community/attacks/xss/
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

I tried the **<b> Deepak </b>** script to make the typed word bold.

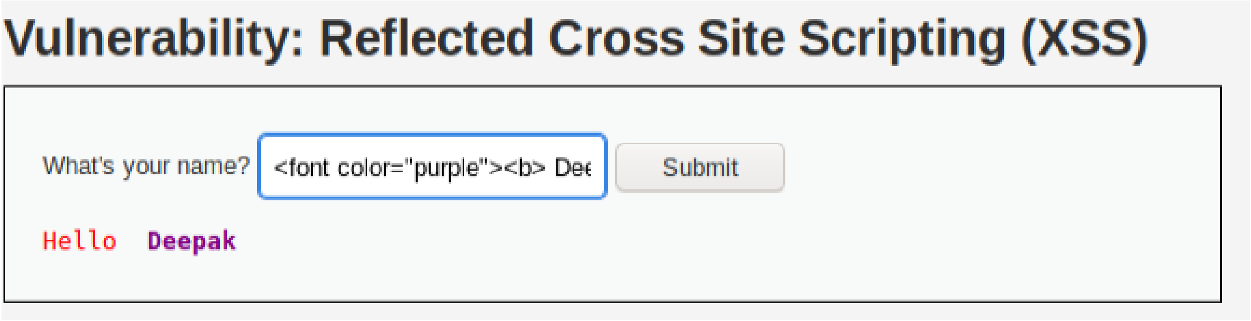


Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? <b> Deepak </b>

Hello **Deepak**

I tried **<font color="purple"><b> Deepak </b></font>** and got the following results.



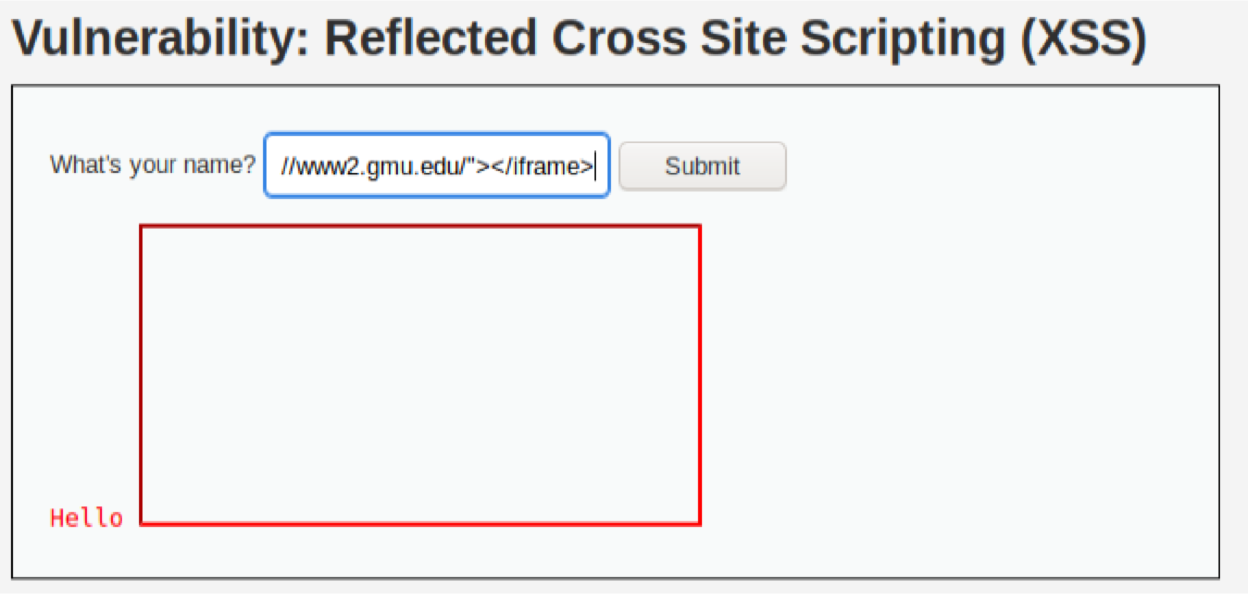
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? <font color="purple"><b> Dee

Hello **Deepak**

Cross-Site Scripting Lab

I tried reflecting gmu website on the frame by using **<iframe src=”https://www2.gmu.edu/”></iframe>** , unfortunately I was not able to see the website loaded on the frame as shown below.

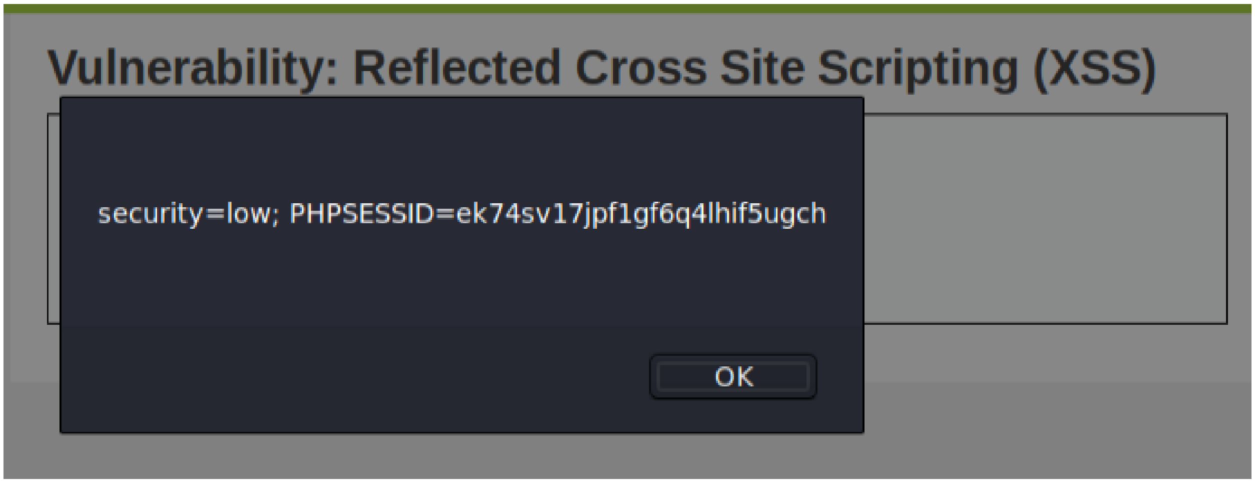


Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? //www2.gmu.edu/"></iframe>

Hello

I used the <script>alert(document.cookie)</script> command in the text box and the output was as follows.



Vulnerability: Reflected Cross Site Scripting (XSS)

security=low; PHPSESSID=ek74sv17jpf1gf6q4lhif5ugch

5. **Conclusion**

   Thus, using the DVWA web application, we were able to use the scripts to inject into the trusted websites. This kind of injection attacks are called Cross-site scripting attacks.