Vulnerability Assessment and Penetration Testing on Web Application

---

Submitted by:

Ansh Sharma

Institute Name:

I.M.S Engineering College

Ghaziabad, Uttar Pradesh, India

Course:

Cybersecurity

Date of Submission:

20th July 2025

Supervisor:

Mr. Hrushikesh Dinkar

Abstract

This project addresses the challenge of securing web applications by demonstrating a systematic Vulnerability Assessment and Penetration Testing (VAPT) process on the Damn Vulnerable Web Application (DVWA). DVWA is an intentionally insecure PHP/MySQL platform used to simulate common web vulnerabilities such as SQL injection, command execution, cross-site scripting (XSS), file inclusion, and more, across varying difficulty levels.

The key features and functionalities of DVWA are then discussed, showcasing its deliberately vulnerable nature for educational purposes. Each attack type, including SQL injection, file upload vulnerabilities, XSS, command execution, and CSRF, is examined in detail, explaining how they can be exploited and the potential risks associated with them.

Mitigation techniques for each vulnerability are explored, emphasizing the significance of secure coding practices, input validation, output encoding, secure file handling, and authentication mechanisms. The report also stresses the importance of proactive security measures, such as regular security assessments, vulnerability scanning, and staying updated with security best practices.

The findings of the report indicate the importance of secure web application development and the practical learning opportunities provided by DVWA. By implementing recommended security measures and adopting secure coding practices, developers can enhance the security of their web applications, protect sensitive data, and mitigate the risks posed by various attacks.

Table of Contents

S No.	CONTENT	Page No.
1	Abstract	2
2	Table of Content	3
3	**Chapter 1 INTRODUCTION**	4
	1.1 Project Overview and Objectives	4
	1.2 Problem Statement	4
	1.3 Approach	5
	1.4 Tools & Techniques	5
4	**Chapter 2 METHODOLOGY**	6
	2.1 Environment Setup	6
	2.2 steps took to exploit vulnerabilities	8
	2.3 Tools & Techniques	8
5	**Chapter 3 Results and Discussion**	9
	3.1 Vulnerabilities Overview Table	9
	3.2 Details of Vulnerabilities Found	10
	3.2.1 Brute Force	10
	3.2.2 Stored XSS	13
	3.2.3 Reflected XSS	15
	3.2.4 File Upload	17
	3.2.5 SQL Injection	19
	3.2.6 File Inclusion	21
	3.2.7 Command Execution	23
	3.2.8 CSRF (Cross-Site Request Forgery)	25
6	**Chapter 4 CONCLUSION**	28
7	**REFERENCES**	29

CHAPTER 1
INTRODUCTION

1.1 Project Overview and Objectives

This project focuses on conducting a Vulnerability Assessment and Penetration Testing (VAPT) exercise using the Damn Vulnerable Web Application (DVWA), a PHP/MySQL web application intentionally designed to be insecure. DVWA replicates real-world vulnerabilities—such as SQL Injection, Cross Site Scripting (XSS),

Command Injection, and Cross Site Request Forgery (CSRF)—across low security levels, offering a safe, legal environment for hands-on cybersecurity training.

- To explain and illustrate the different types of attacks that can be performed on DVWA, specifically focusing on SQL injection, file upload, cross-site scripting (XSS), command execution, and cross-site request forgery (CSRF) attacks.
- To explore the vulnerabilities associated with each attack type, including their potential risks, impact, and consequences.
- To showcase real-world examples and demonstrations of these attacks within the DVWA environment, helping readers grasp the practical aspects of the vulnerabilities and their exploitation.
- To discuss and recommend effective mitigation techniques and best practices for preventing and mitigating the identified vulnerabilities.
- To emphasize the importance of secure web application development and raise awareness about the need for proactive measures to enhance web application security.

1.2 Problem Statement

Web applications frequently suffer from common vulnerabilities highlighted in the OWASP Top 10. In real-world development environments, such weaknesses can lead to data breaches, defacements, or full system compromise. DVWA serves as a practical learning platform, enabling users to experience exploitation techniques directly in a controlled setting. The core problem addressed is the gap between theoretical knowledge and actual exploitation skills—this project aims to bridge that gap by simulating attacks in a realistic lab setup.

1.3 Approach

In this project we will simulate a full Vulnerability Assessment and Penetration Testing (VAPT) workflow against the Damn Vulnerable Web Application (DVWA).

Steps include:

1. Deploying DVWA in a safe, isolated environment like Kali Linux VM .
2. Manual exploitation of critical vulnerabilities (e.g., SQL Injection, Command Injection, XSS, CSRF) using Burp Suite, custom payloads.
3. Risk evaluation, crafting proof of concept attacks, assessing severity, and documenting findings.
4. Recommendation , outlining mitigation strategies aligned with best coding practice.

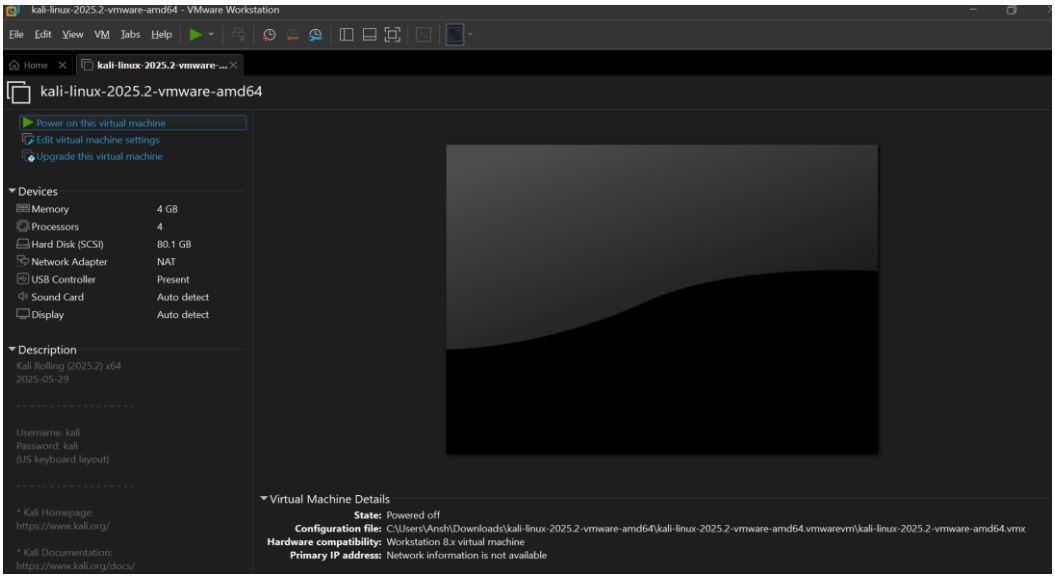
1.4 Tools & Techniques

- **Burp Suite (PortSwigger):** Web traffic capture and manipulation tool used for payload testing and analysis.
- **Kali Linux (Offensive Security):** Penetration testing OS running on VMware for testing DVWA.
- **Metasploitable2 :** Vulnerable VM hosting DVWA for security testing.
- **VMware Workstation:** Virtualization platform used to host testing VMs in an isolated network.

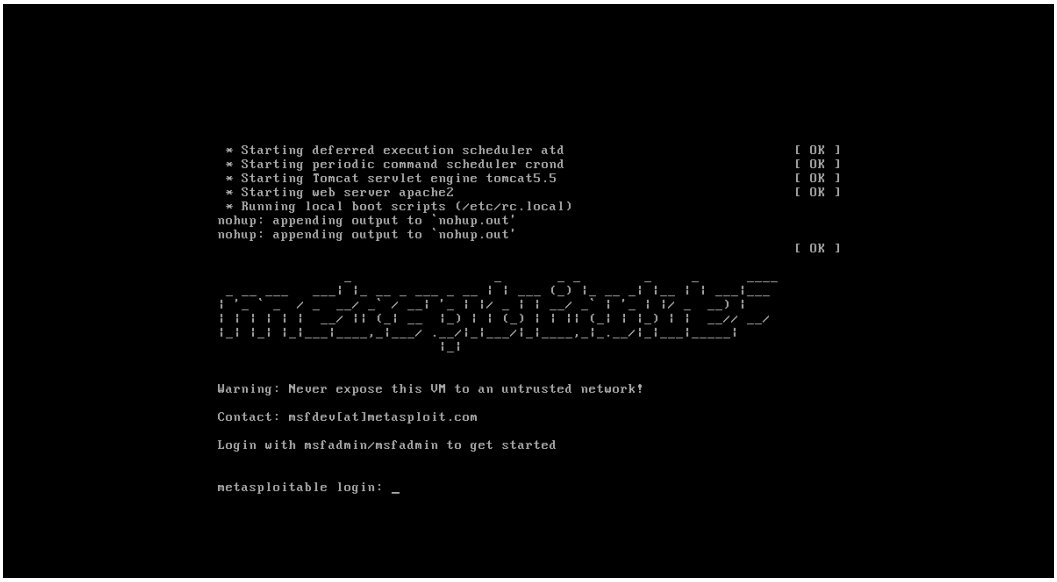
CHAPTER 2
METHODOLOGY

2.1 Setup the Environment

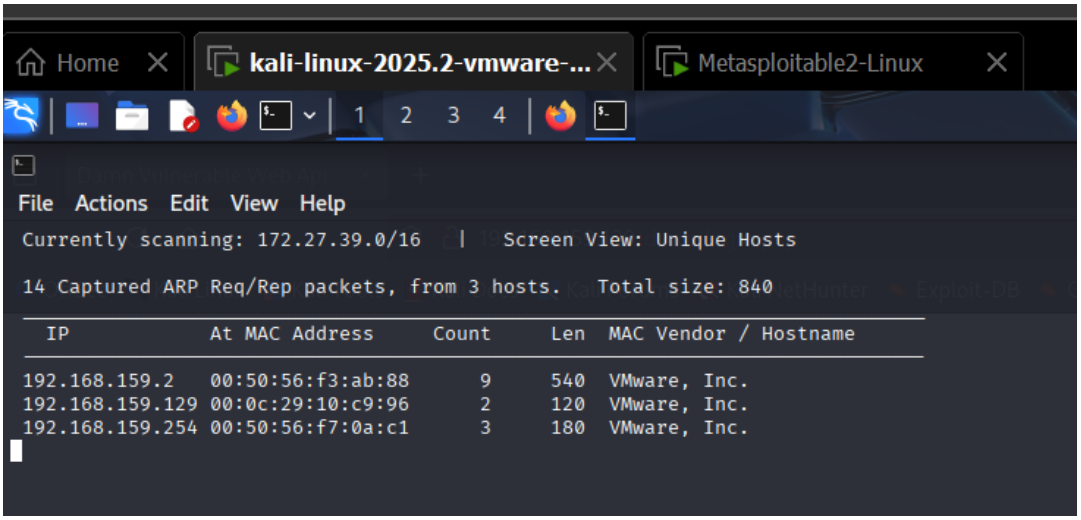
- **Install a virtualization platform:** To run Metasploitable2, you will need a virtualization platform such as Oracle VirtualBox or VMware Player. Download and install the virtualization software of your choice.



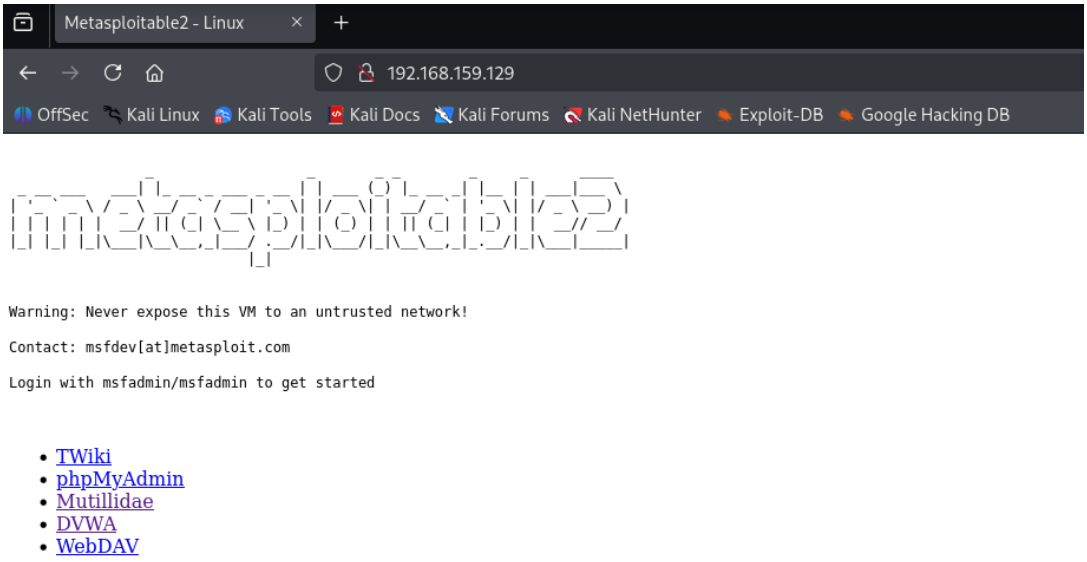
- **Obtain Metasploitable2:** Metasploitable2 is a vulnerable Linux virtual machine designed for testing purposes. You can download the Metasploitable2.
- **Import the Metasploitable2 :** Open your virtualization software and import the downloaded Metasploitable2 then Launch the Metasploitable 2 on your VM.



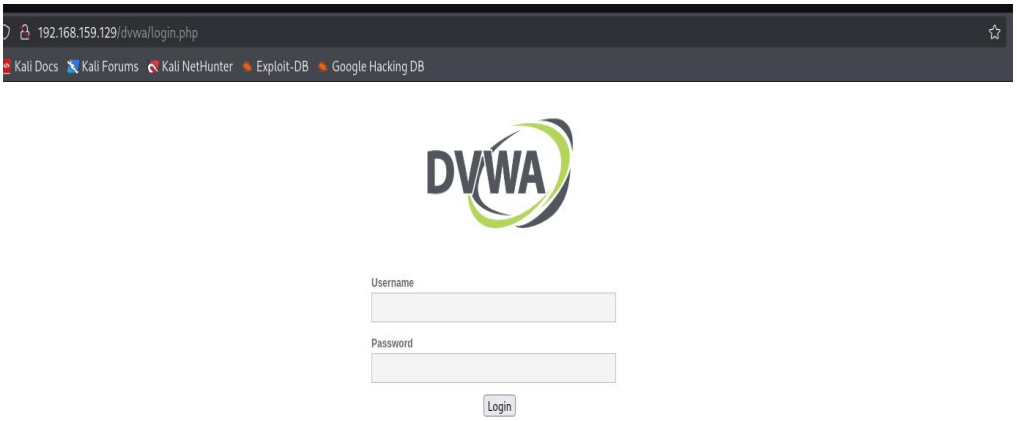
Netdiscover



DVWA



Default username=”admin” and password=”password”



Test Each Vulnerability in Low Security

2.2 steps took to exploit vulnerabilities.

- **Burp Suite – Repeater & Intruder:**
- **SQL Injection:** Tested payloads like 1' OR '1'='1' -- and UNION SELECT user, password FROM users -- Payloads manipulated ID or username fields to test injection viability.
- **XSS:** Injected alert('XSS') (Low) and altered variants for Medium/High levels in Name or Message fields to test stored XSS behavior.
- **Brute Force:** Employed Intruder with wordlists to attack login form; used Comparer and response-length sorting to detect successful logins.
- **Command Injection:** Entered commands like ; whoami into Ping module to validate OS command execution.
- **File Upload:** Uploaded backdoor.php(ansh.php) to test for remote code execution.
- **CSRF:** Crafted requests to change passwords without CSRF

2.3 Tools & Techniques

- **Burp Suite (PortSwigger):** Web traffic capture and manipulation tool used for payload testing and analysis.
- **Kali Linux (Offensive Security):** Penetration testing OS running on VMware for testing DVWA.
- **Metasploitable2 (Rapid7):** Vulnerable VM hosting DVWA for security testing.
- **VMware Workstation:** Virtualization platform used to host testing VMs in an isolated network.

CHAPTER 3
RESULT AND DESCUSSION

3.1 Vulnerabilities Overview Table

S No.	Vulnerability	Severity	Risk score
1	Brute Force	High	7.0
2	Stored XSS	High	7.0
3	Reflected XSS	Medium	6.5
4	File Upload	Critical	9.0
5	SQL Injection	High	7.5
6	File Inclusion	Critical	9.2
7	Command Execution	High	8.6
8	CSRF (Cross-Site Request Forgery)	Medium	5.5

Severity Summary

- **High (7.0–8.9):** SQL Injection (Classic), Brute Force , Stored XSS, Command Injection
- **Critical (9.0+):** File Upload , Command Execution
- **Medium (4.0–6.9):** , Reflected XSS, CSRF

Notes

- High and Critical severity findings pose immediate risk and demand urgent remediation.
- Medium severity items are less impactful but remain exploitable and should be addressed promptly.

3.2Details of Vulnerabilities Found

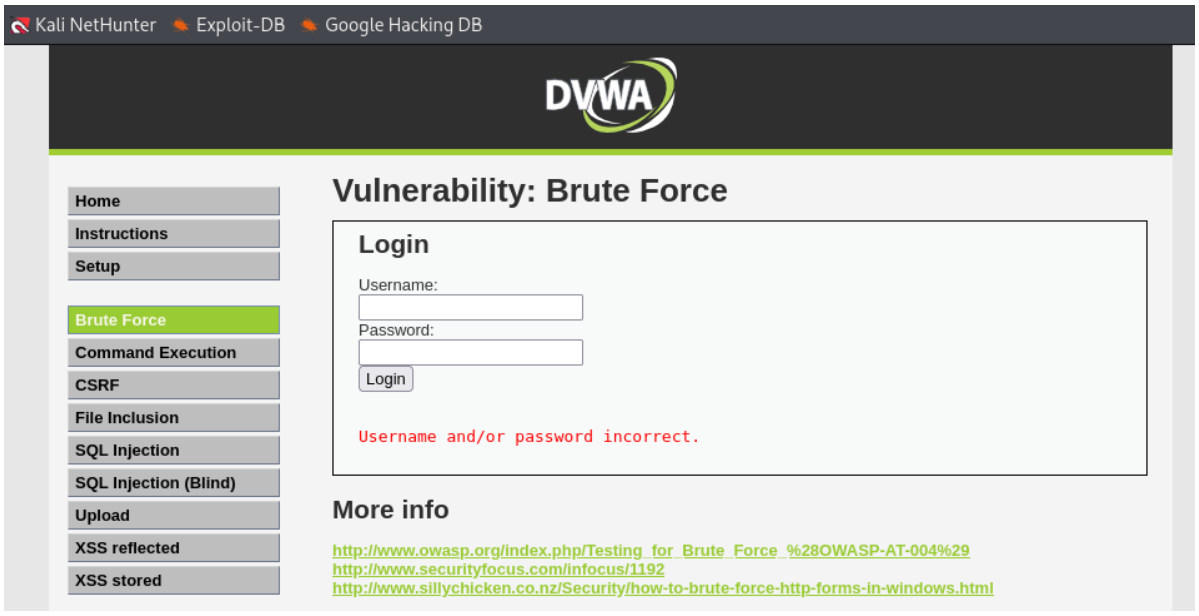
3.2.1 Brute Force

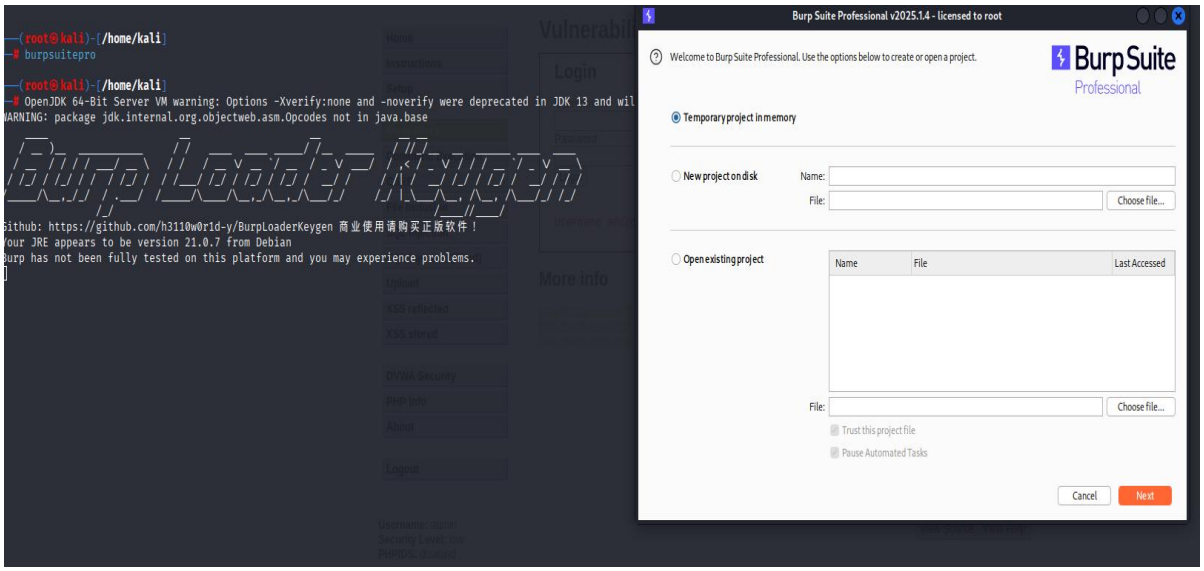
Brute force attacks involve systematically attempting multiple combinations of usernames and passwords to gain unauthorized access to accounts.

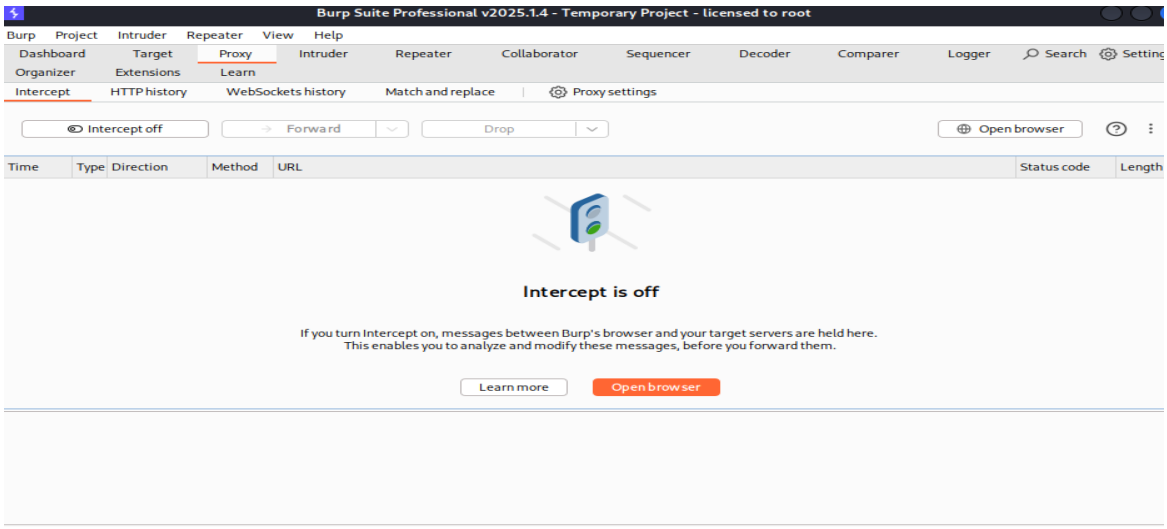
Severity : High

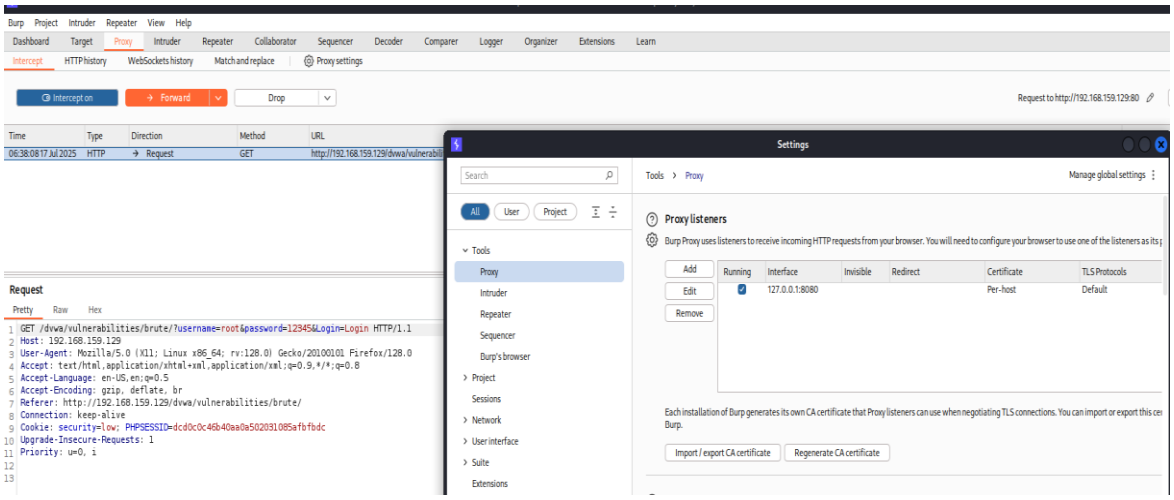
Impact: Successful brute-force attacks can compromise user accounts, leading to unauthorized access to sensitive information or control over the user's actions within the application.

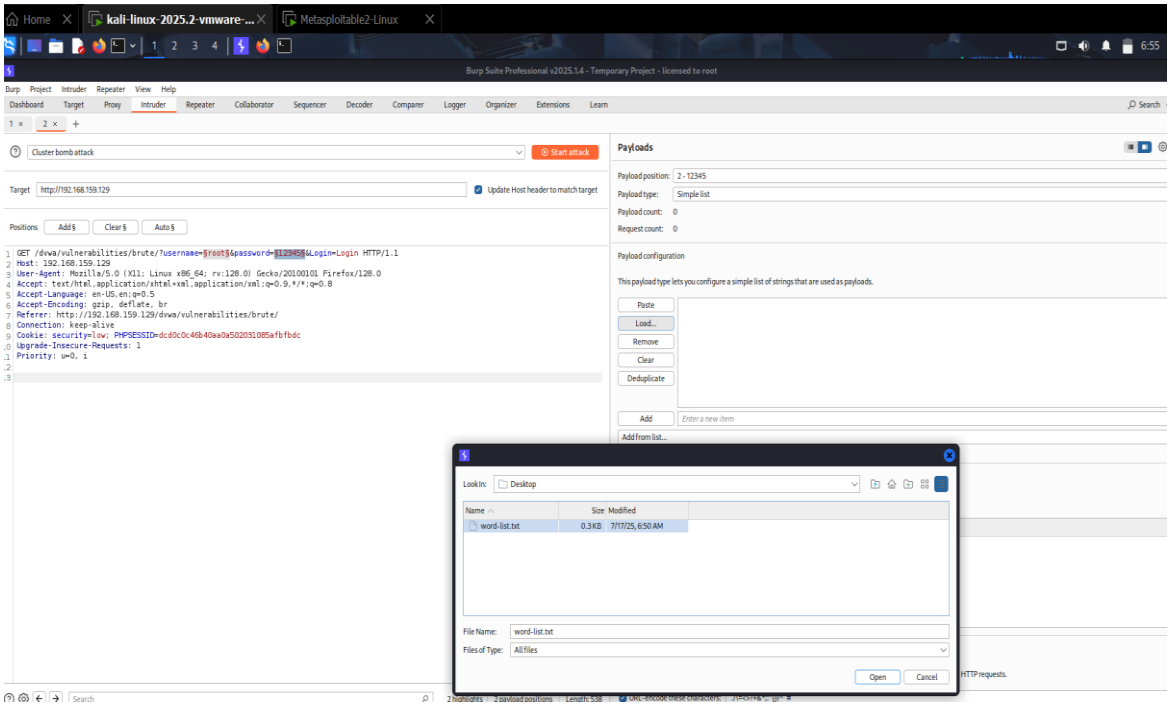
Proof of Concept (PoC):

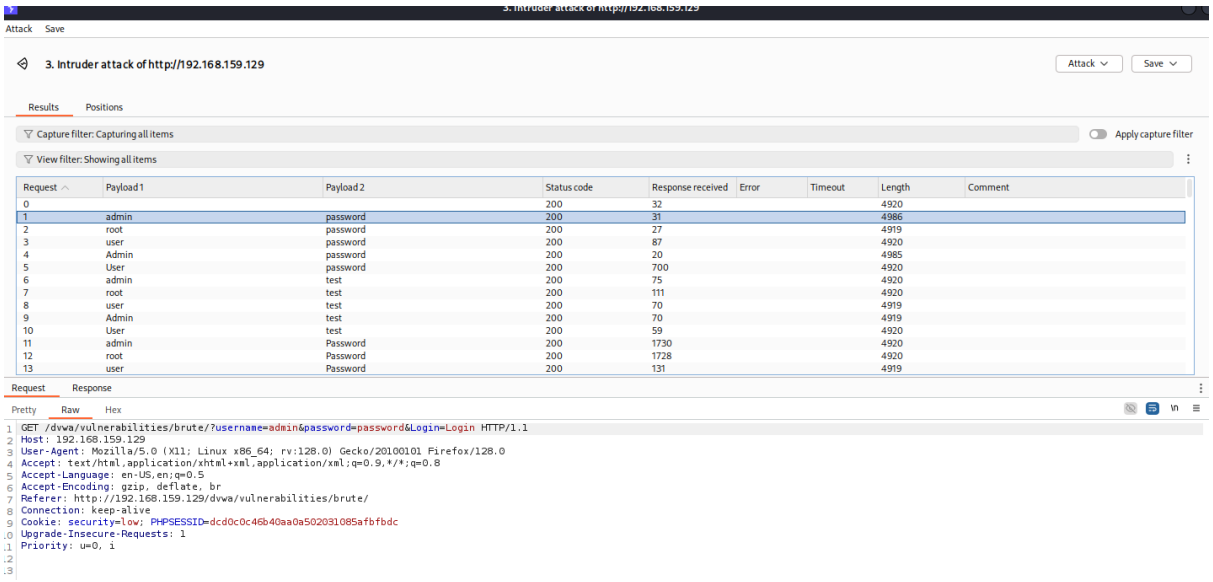





Burp Suite Professional v2025.1.4 - Temporary Project - licensed to root
Intercept off
Forward
Drop
Open browser
Intercept is off
If you turn Intercept on, messages between Burp's browser and your target servers are held here.
This enables you to analyze and modify these messages, before you forward them.
Learn more
Open browser


Settings
Proxy listeners
Request
GET /dvwa/vulnerabilities/brute/?username=root&password=12345&Login=Login HTTP/1.1


Burp Suite Professional v2025.1.4 - Temporary Project - licensed to root
Payloads
GET /dvwa/vulnerabilities/brute/?username=$root$&password=12345&Login=Login HTTP/1.1



Mitigations:

- Implement rate limiting, account lockouts, or exponential backoff delays after multiple failed login attempts.
- Enable Multi Factor Authentication (MFA) to make stolen credentials less useful.

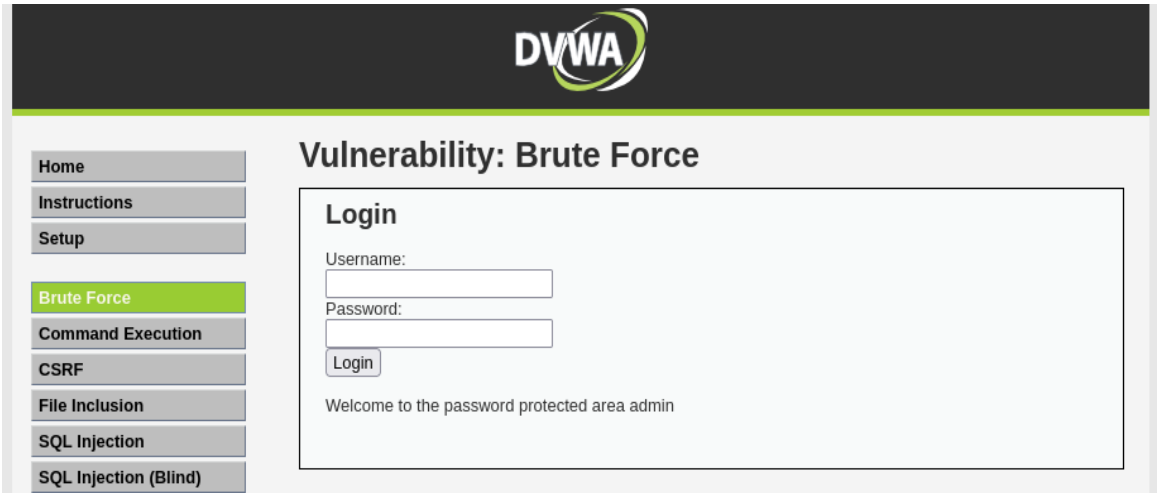
3.2.2 XSS Stored

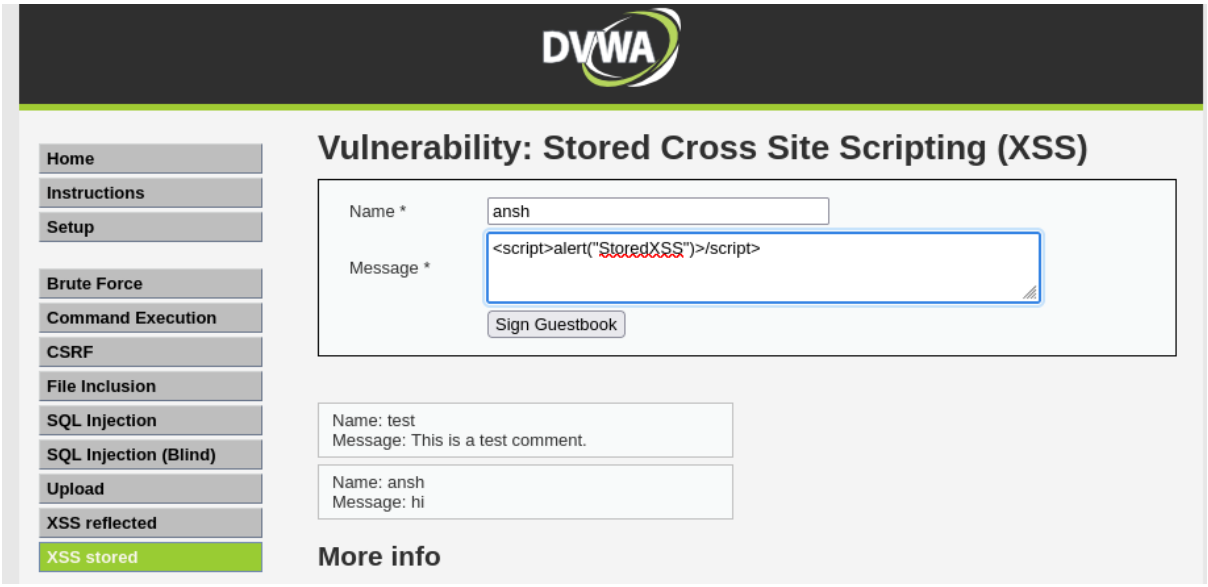
Stored XSS, also known as persistent XSS, occurs when an attacker injects malicious scripts into a website's database. When a user visits a page with this injected script, the script runs in their browser, potentially stealing sensitive information like cookies or session tokens or performing actions on behalf of the user without their consent.

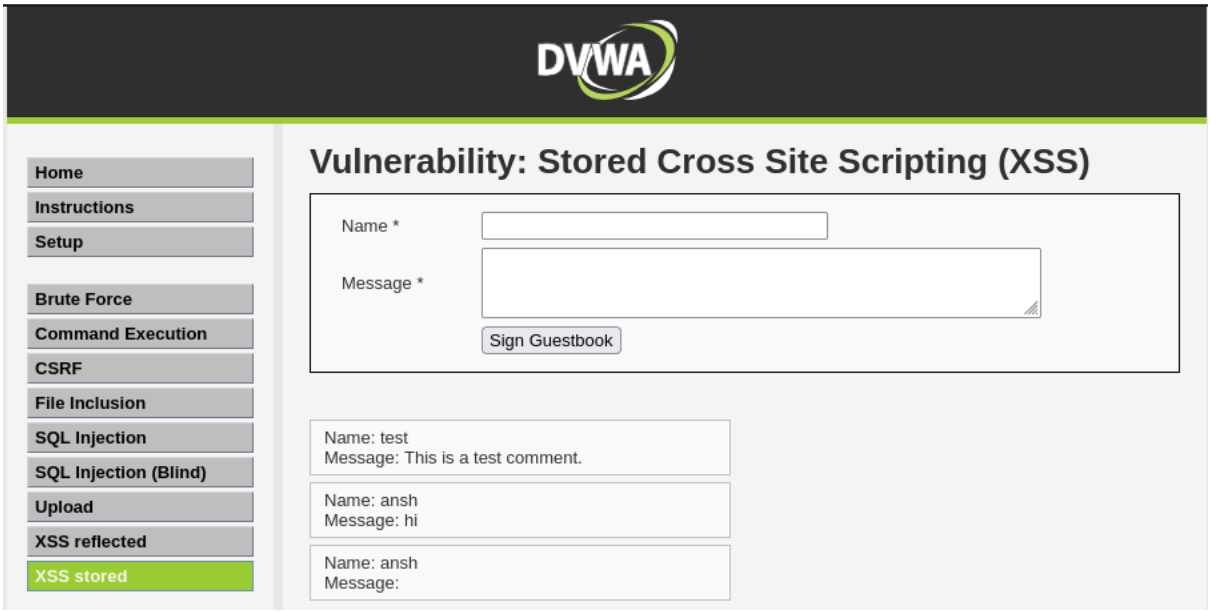
Severity: High

Impact: Once the malicious script is executed by a victim, it can lead to session hijacking, redirection to malicious sites, or other harmful activities, significantly affecting user trust and security

Proof of Concept (PoC):







After submitting the XSS payload. Whenever a user visits this page, the malicious script executes, proving a successful Stored XSS attack.

Mitigations:

- Validate inputs to allow only expected values, eliminating malicious scripts.
- Escape and encode outputs depending on the context (HTML, JavaScript, CSS) so user data isn't executed by the browser.
- Deploy a Content Security Policy (CSP) to restrict script sources and block inline/external attacks.

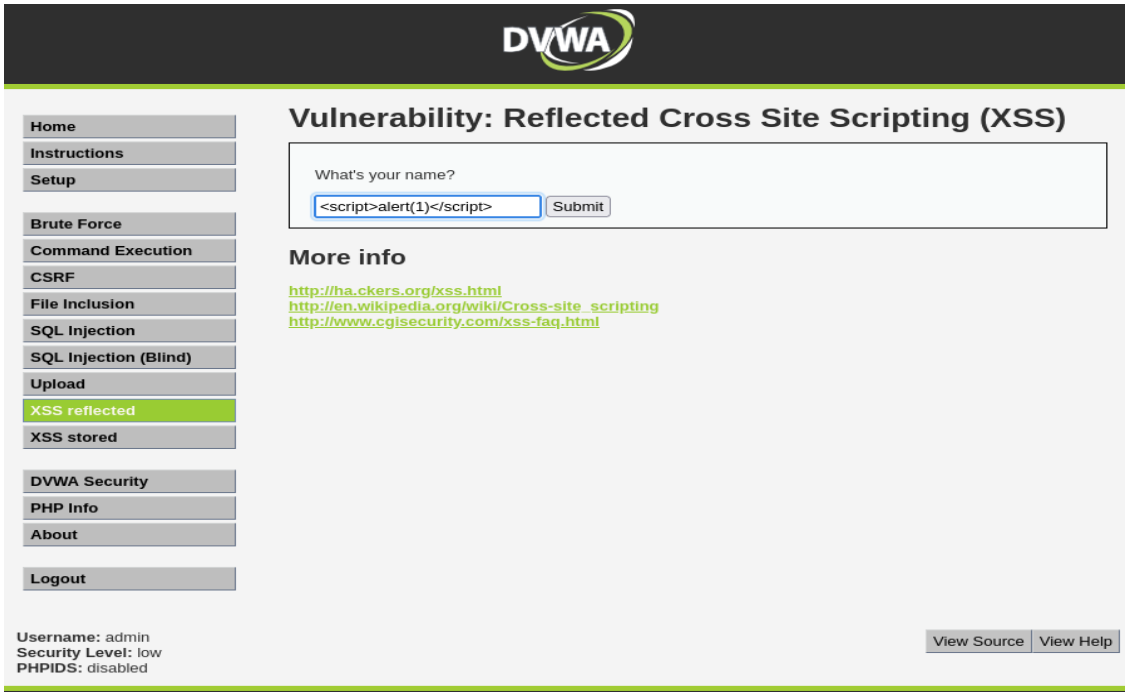
3.2.3 XSS Reflected

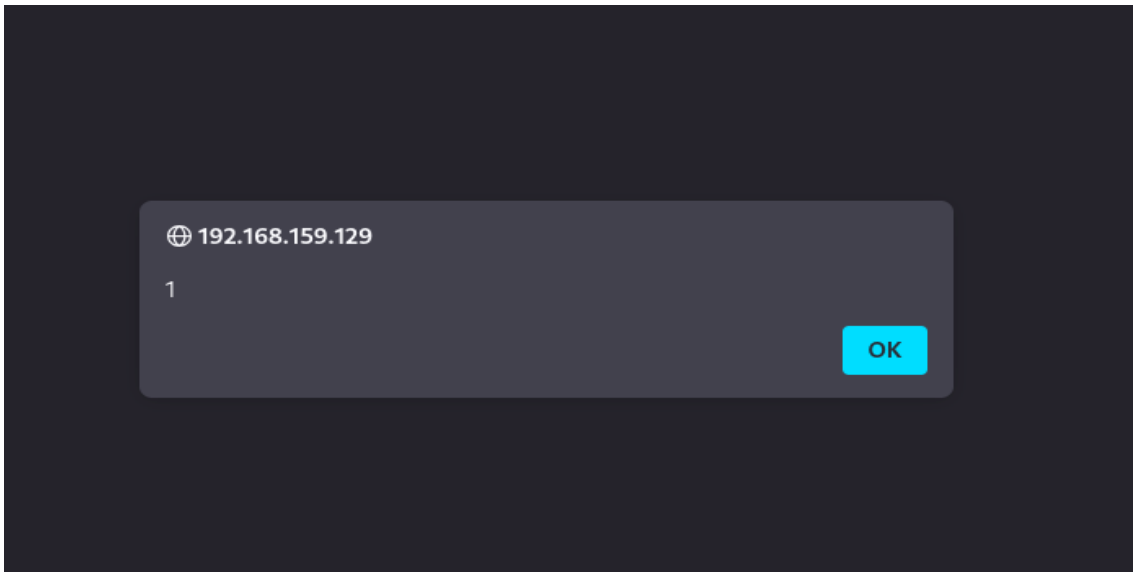
Reflected XSS occurs when an attacker sends a malicious script as part of a URL or form submission. If the server reflects this input back to the user without proper validation or escaping, the script can be executed in the user's browser.

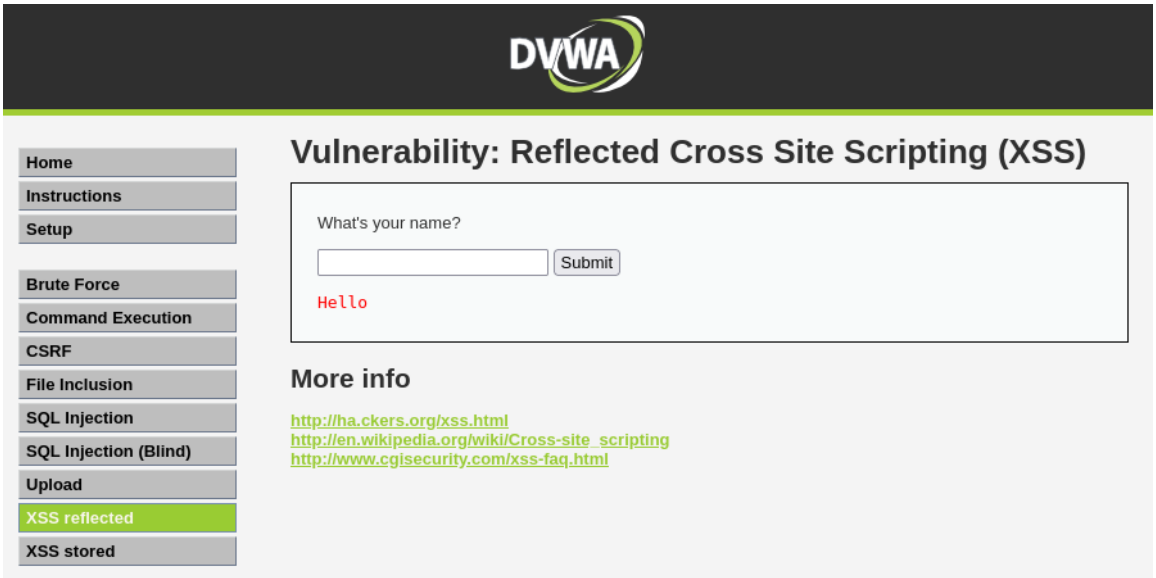
Severity: Medium

Impact: The impact is similar to other XSS types, allowing attackers to execute scripts in a user's browser context, potentially stealing information or performing actions without the user's consent.

Proof of Concept (PoC):







Mitigations:

Same as stored XSS: strong input validation, output encoding, and use of a strict CSP to stop injected payloads from running in the browser.

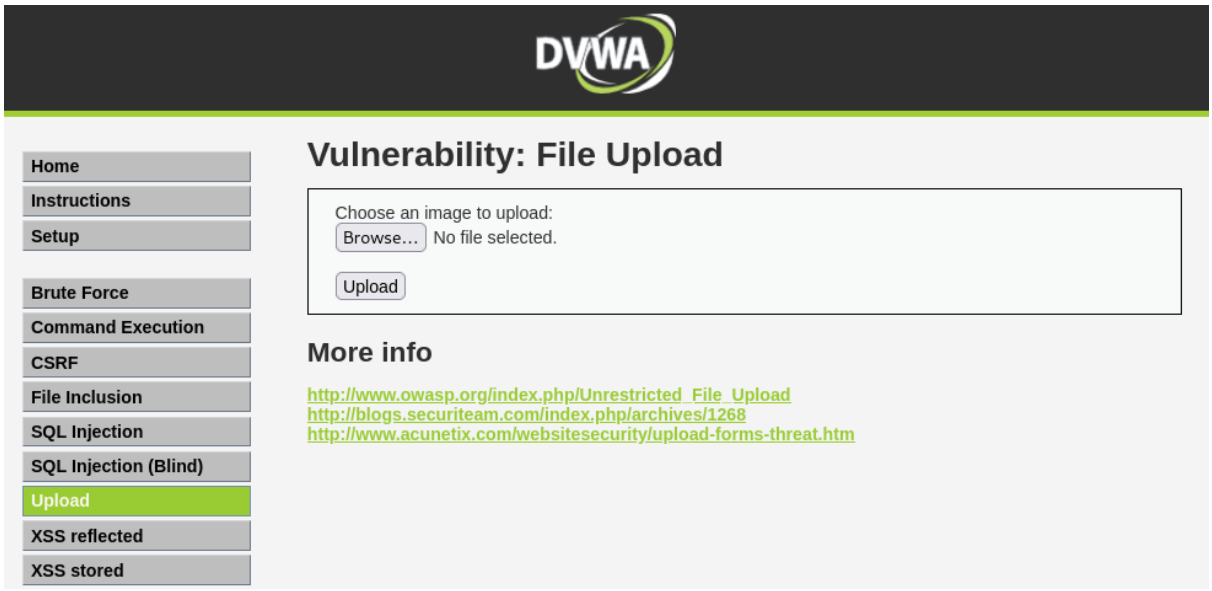
3.2.4 File Upload Vulnerability

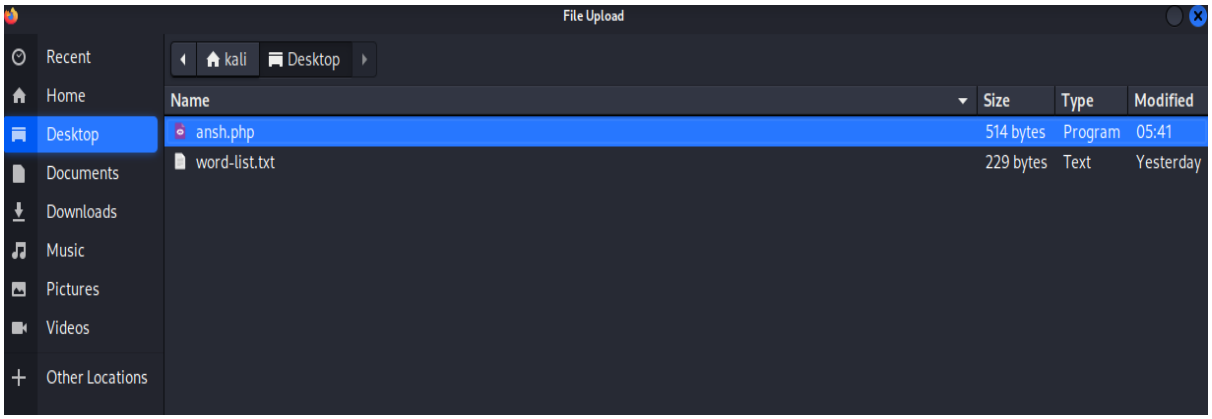
File upload vulnerabilities occur when an application allows users to upload files without proper validation or restriction, leading to the potential execution of malicious files (e.g., PHP shells) on the server.

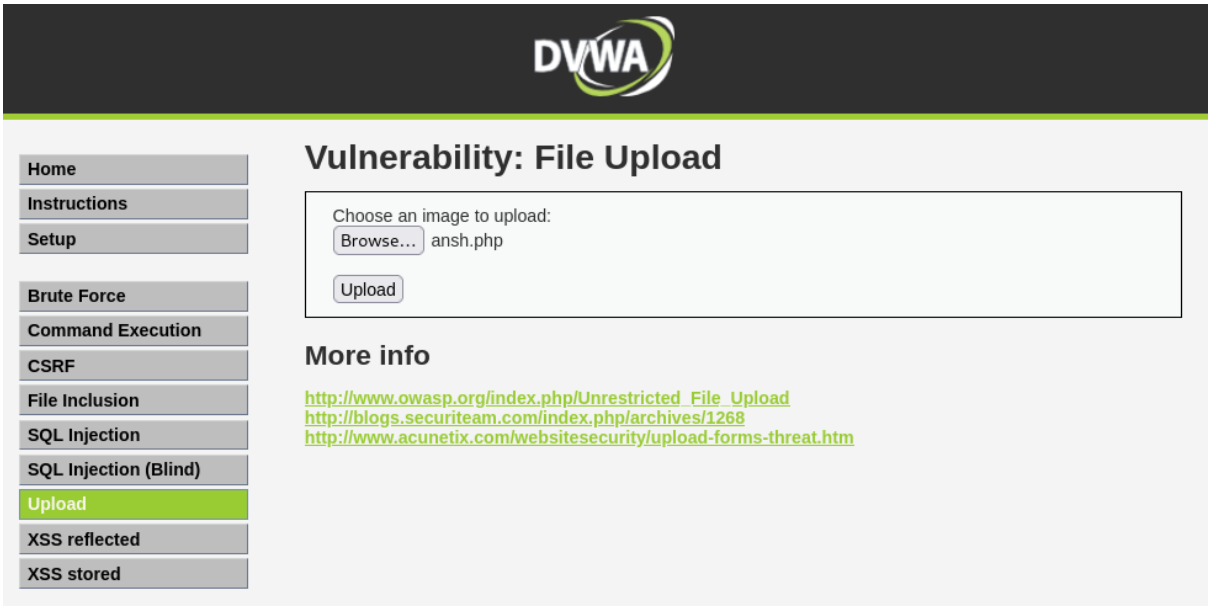
Severity: Critical

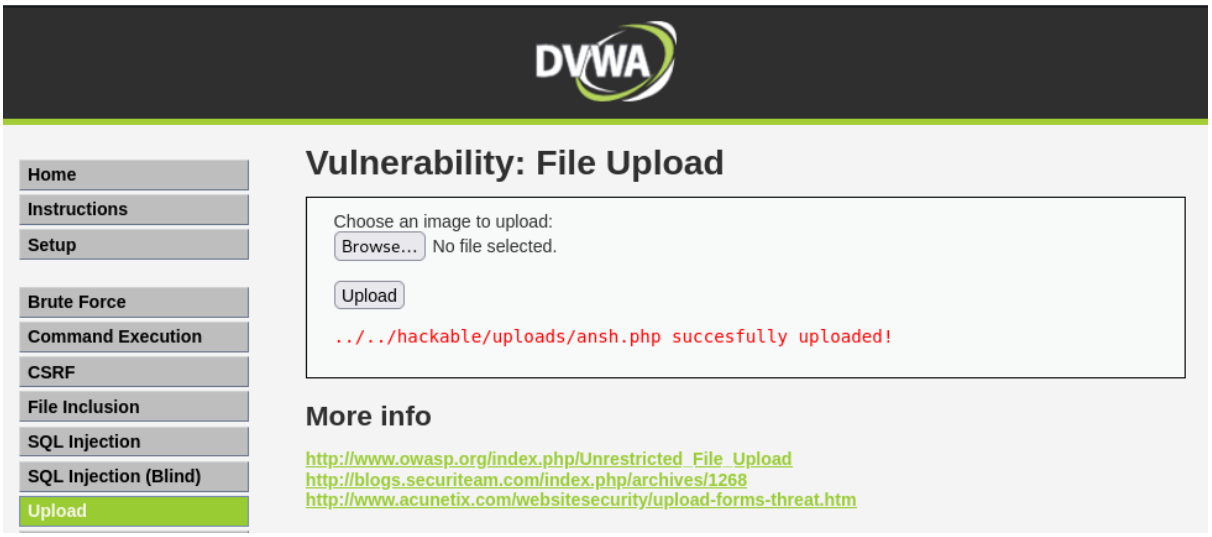
Impact: If exploited, this vulnerability can lead to server-side code execution, allowing attackers to gain unauthorized access to the server, modify data, or even take complete control of the server.

Proof of Concept (PoC):









Mitigations:

- Allowlist file types (e.g. .jpg, .png, .pdf), check MIME types and file “magic bytes” to validate uploads.
- Rename uploaded files using random IDs or hashes, not user-provided names.
- Scan files with antivirus or malware detection, and sanitize metadata or embedded scripts

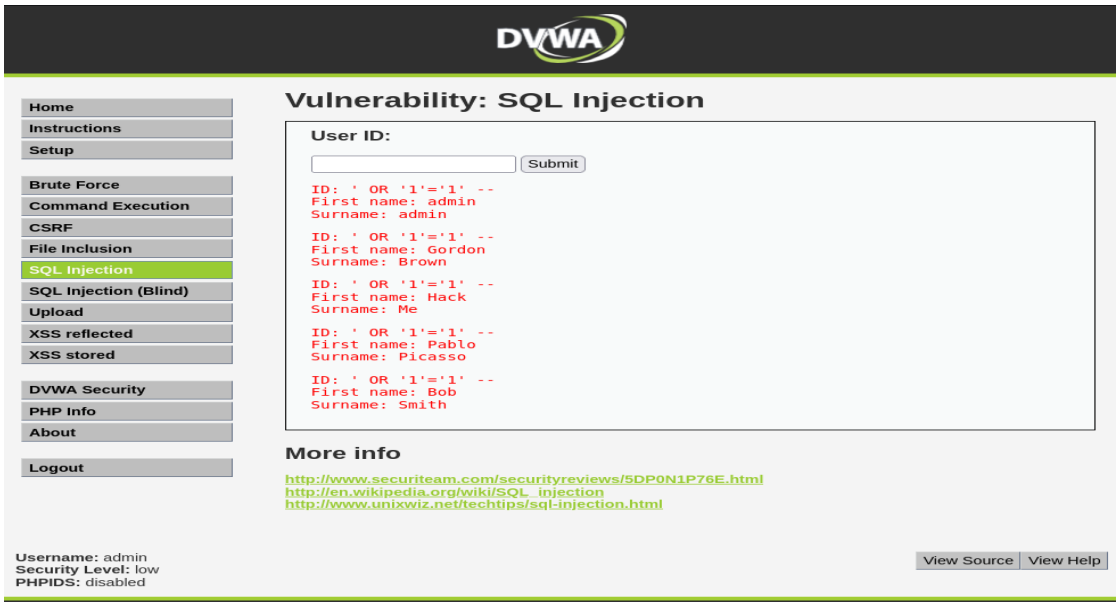
3.2.5 SQL Injection

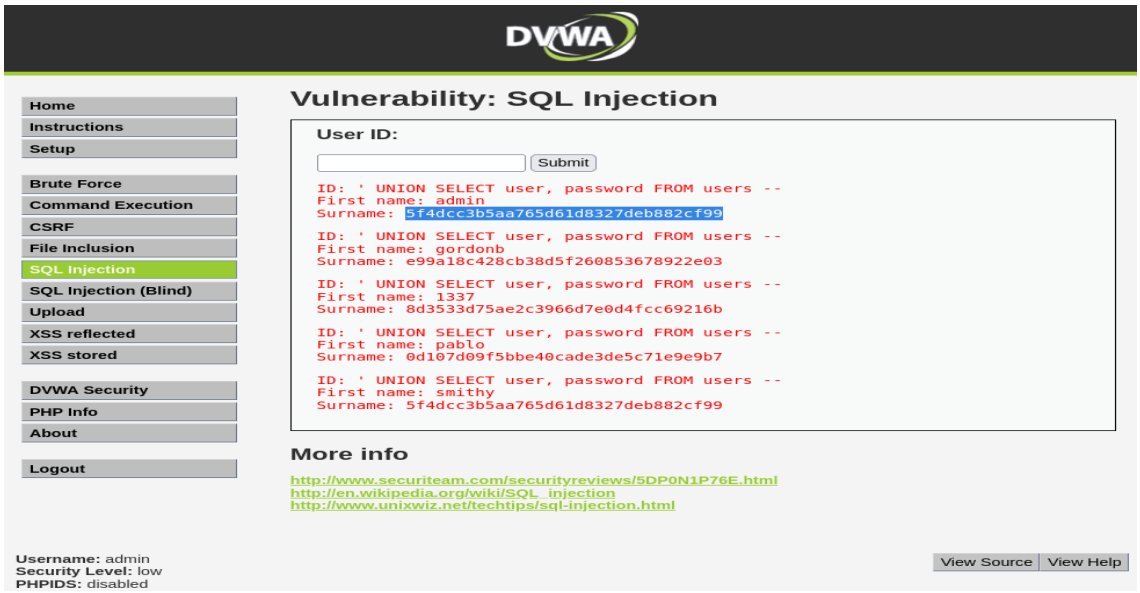
It is a type of attack in which an attacker injects malicious code into a website's SQL statement and gains access to sensitive information or performs malicious actions on the database. This is typically done by manipulating input fields in a web application that is connected to a database, such as a login form or a search box, in such a way as to trick the application into executing unintended SQL commands

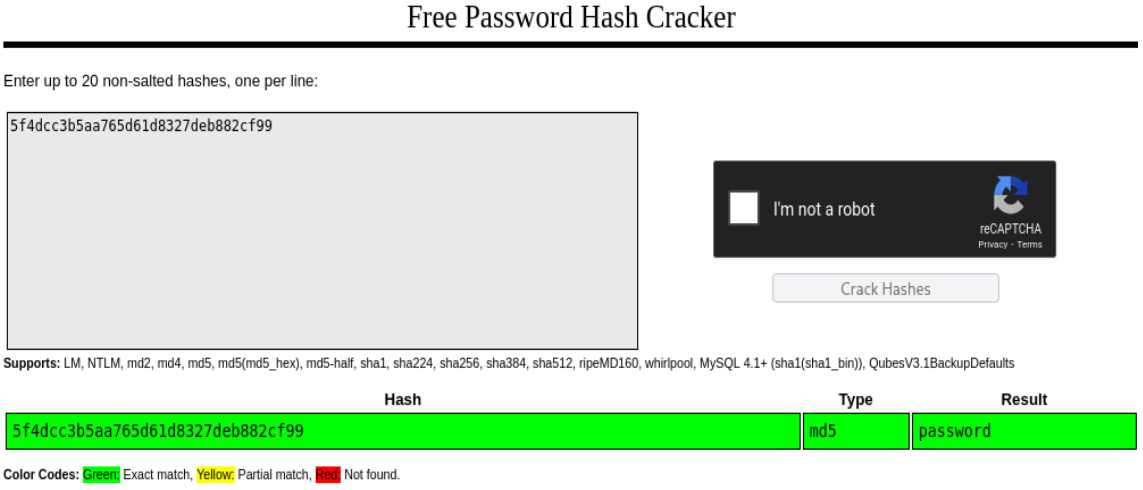
Severity: High

Impact: SQL injection attacks can allow attackers to bypass authentication, access, modify, or delete sensitive data, or even execute commands on the operating system. They can also be used to create new user accounts with high privileges or to perform other malicious action.

Proof of Concept (PoC):







Mitigations:

- Use parameterized queries or prepared statements; never build SQL via unchecked string concatenation.
- Deploy a Web Application Firewall (WAF) to block common injection patterns as an additional layer

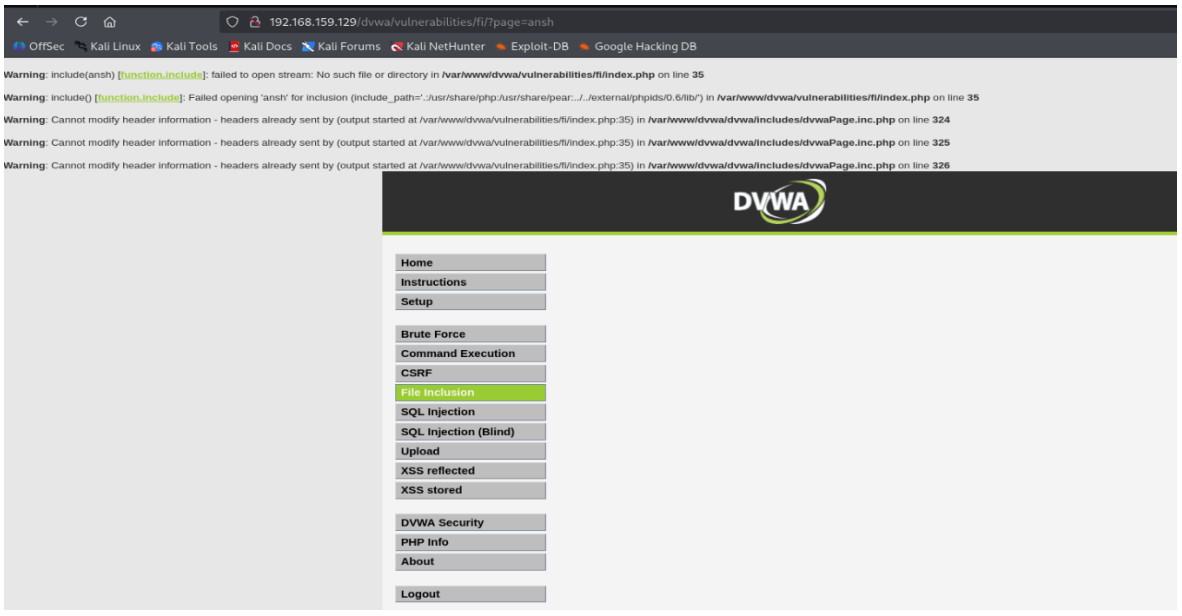
3.2.6 File Inclusion

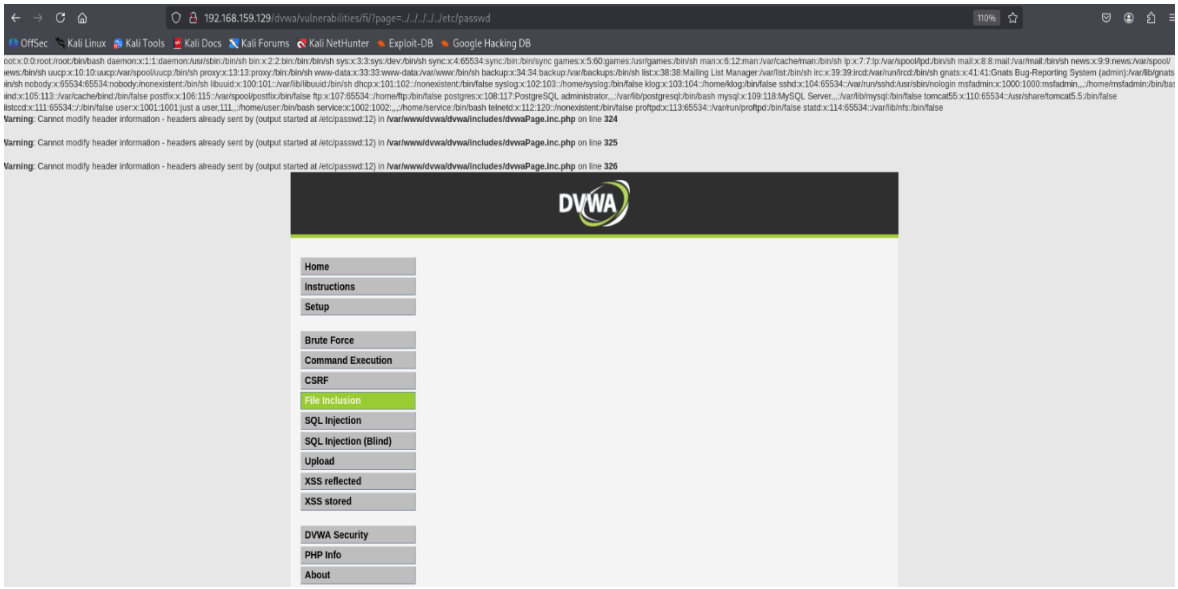
File inclusion vulnerabilities occur when an application dynamically includes files in a way that allows an attacker to specify a file path, potentially leading to arbitrary code execution or file access.

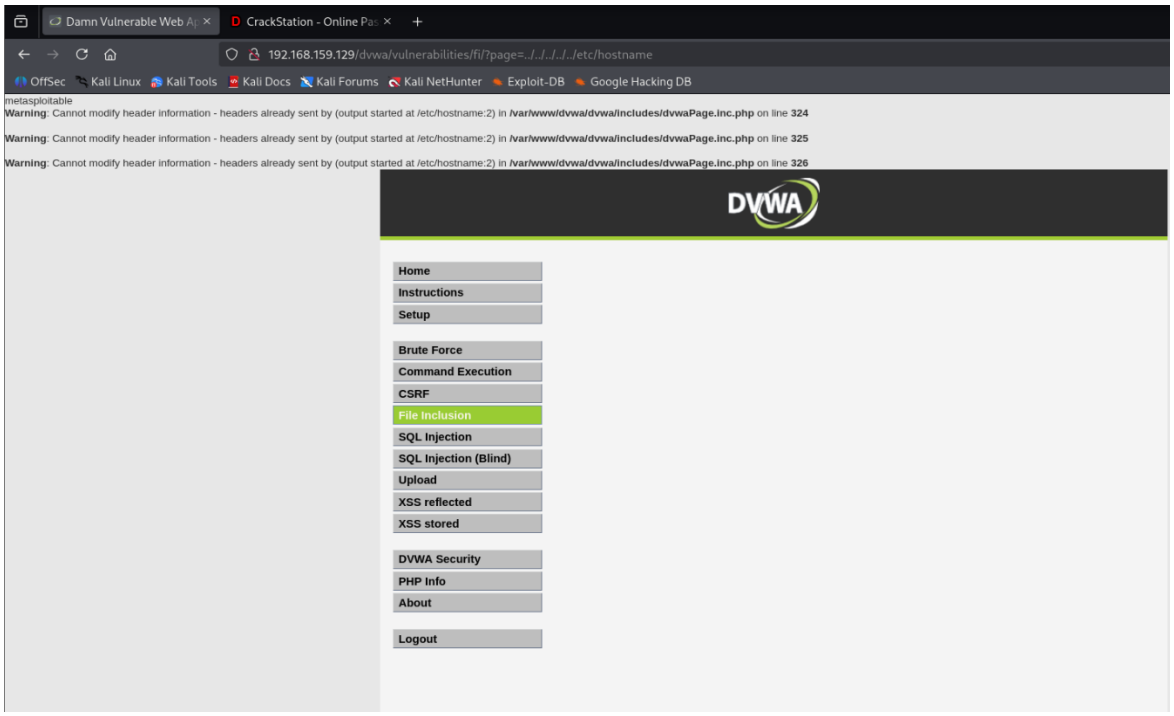
Severity: Critical

Impact: Depending on the level of access obtained, attackers can execute arbitrary code, access sensitive files, or even gain full control over the server.

Proof of Concept (PoC):







Mitigations:

- Ensure any file path input uses a strict allowlist—never dynamically include files based on user input.
- Validate and sanitize paths, avoiding directory traversal (..) or remote fetches.
- Run inclusions in restricted or sandboxed environments to prevent execution of unexpected files.

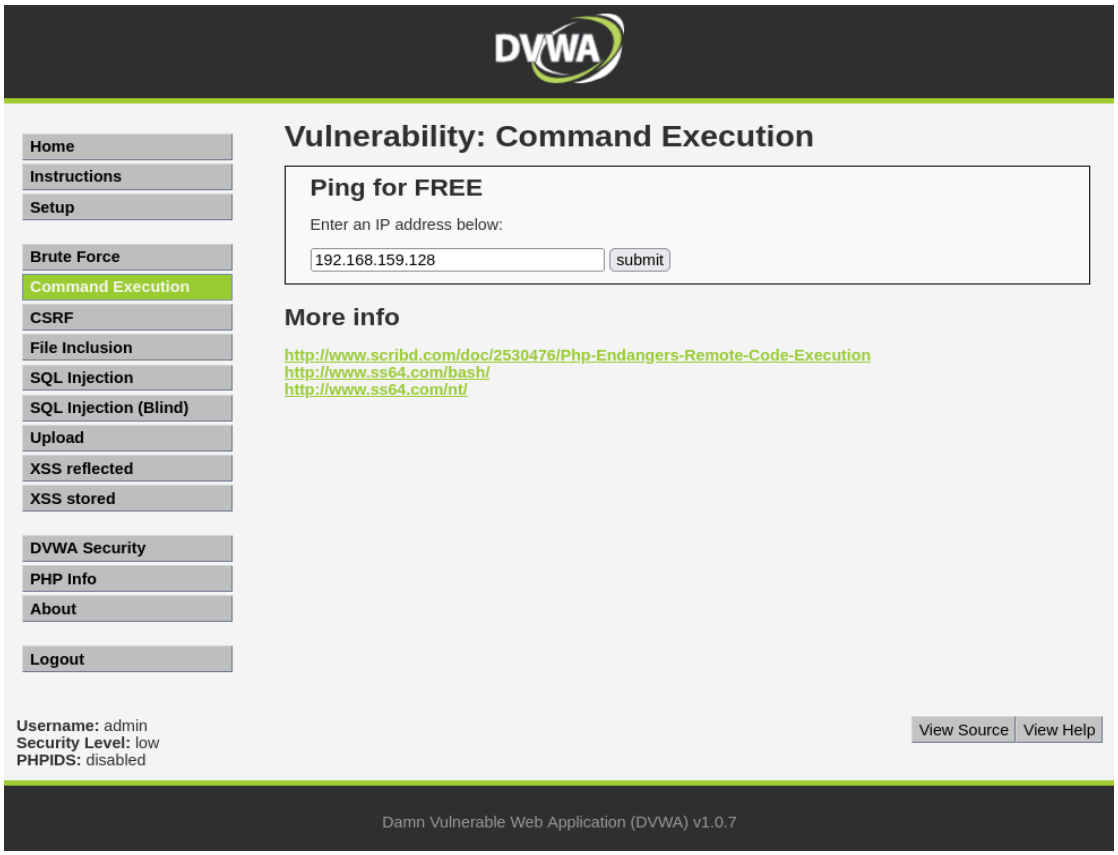
3.2.7 Command Execution

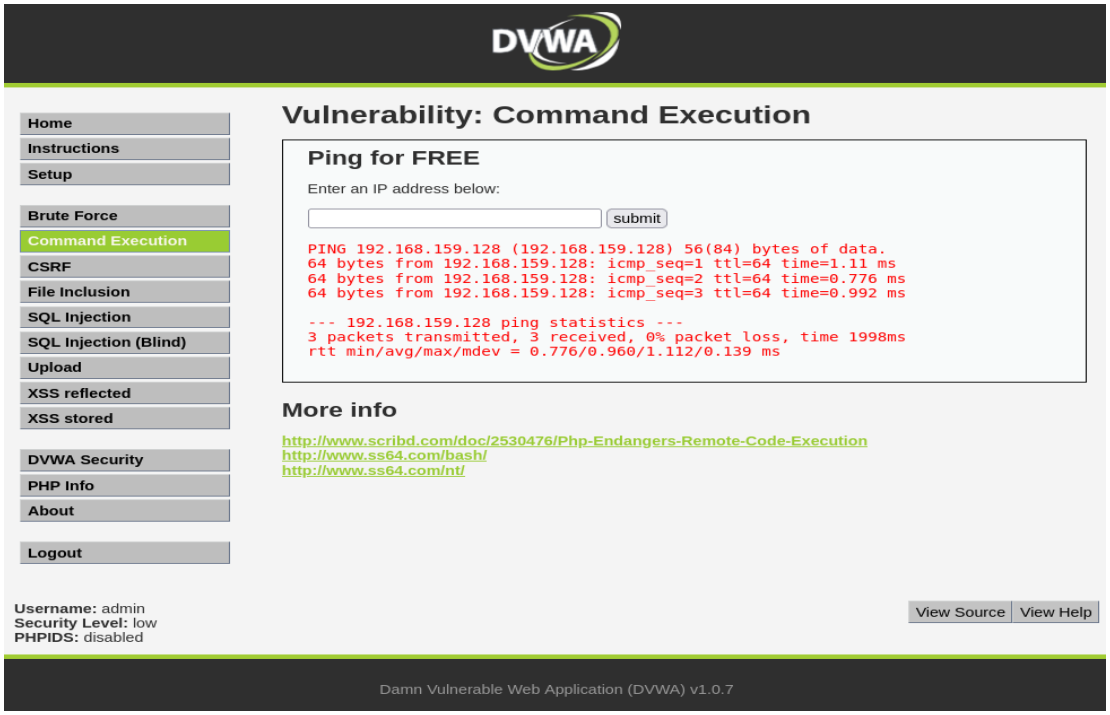
Command Execution vulnerabilities occur when an application directly executes system-level commands based on user input without proper sanitization or validation. This flaw allows attackers to execute arbitrary commands on the server, potentially compromising its security and integrity.

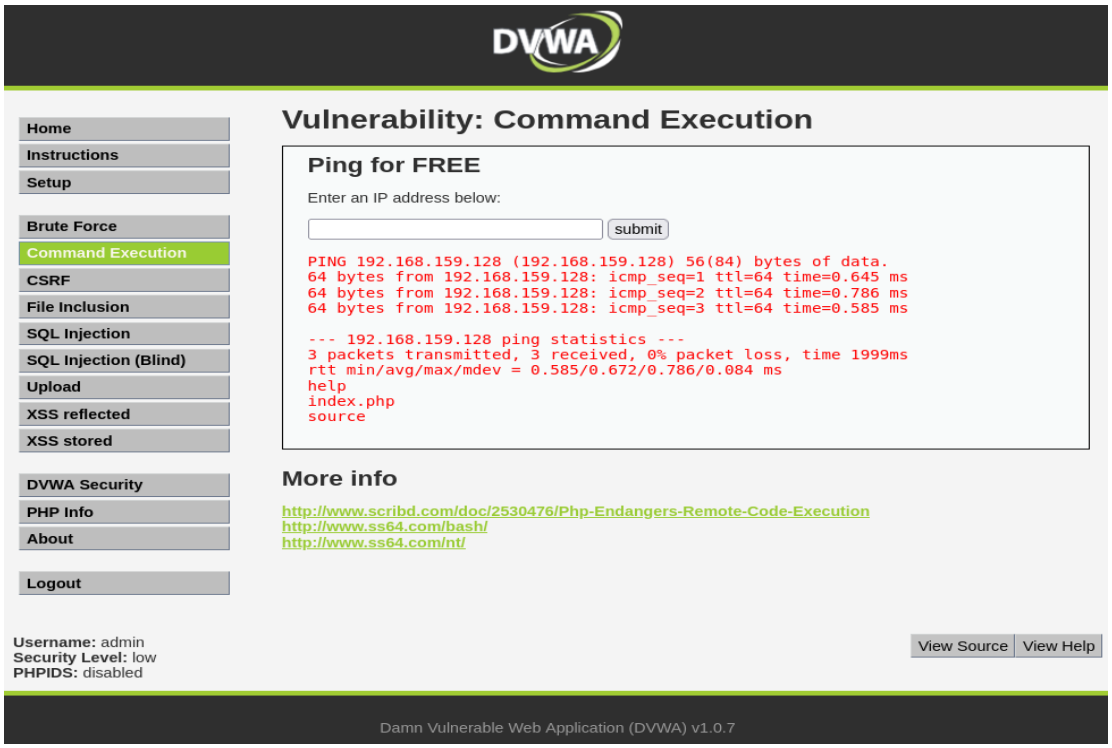
Severity: HIgh

Impact: Successful exploitation of this vulnerability can provide an attacker with the ability to run any command on the server, leading to a wide range of potential consequences. This can include reading sensitive data, modifying or deleting files, installing malicious software, or even gaining a persistent foothold on the server, potentially leading to full system compromise.

Proof of Concept (PoC):







Mitigations:

- Avoid shell calls with user supplied input. If unavoidable, use strict input validation and parameterized execution functions.
- Run commands under least privilege, and isolate them if possible.
- Escape or sanitize inputs carefully, avoiding injection via special characters.
- Monitor execution logs and limit repeated failed or suspicious commands.

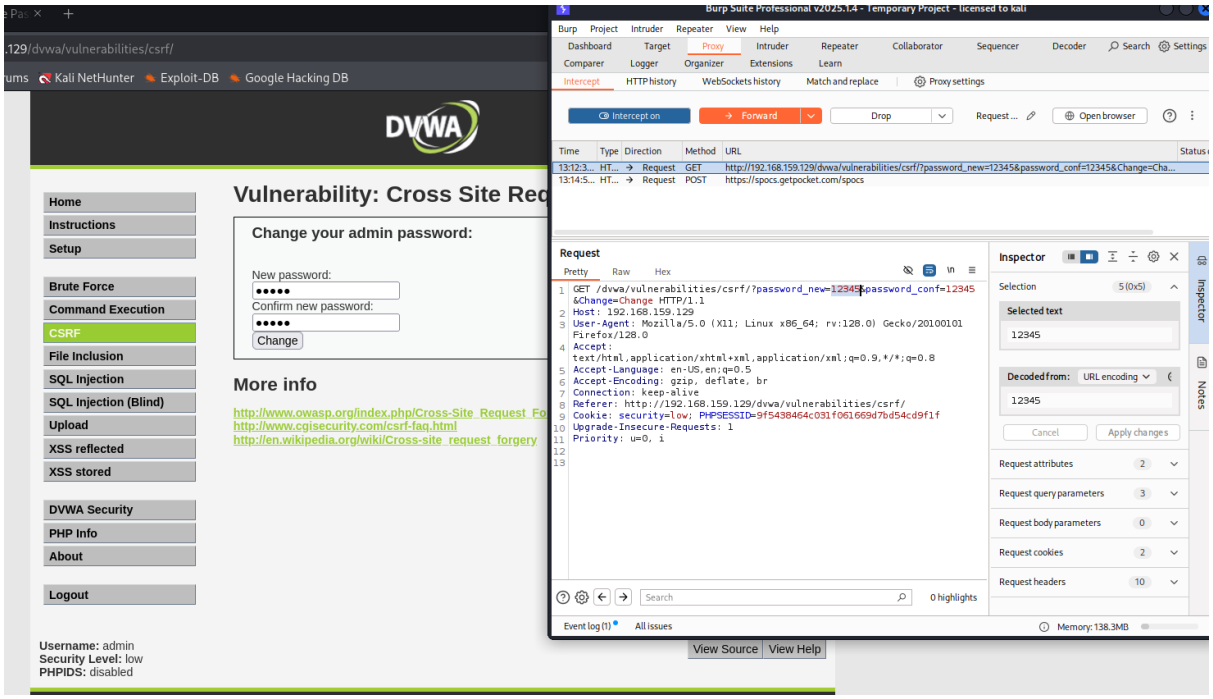
3.2.8 CSRF (Cross-Site Request Forgery)

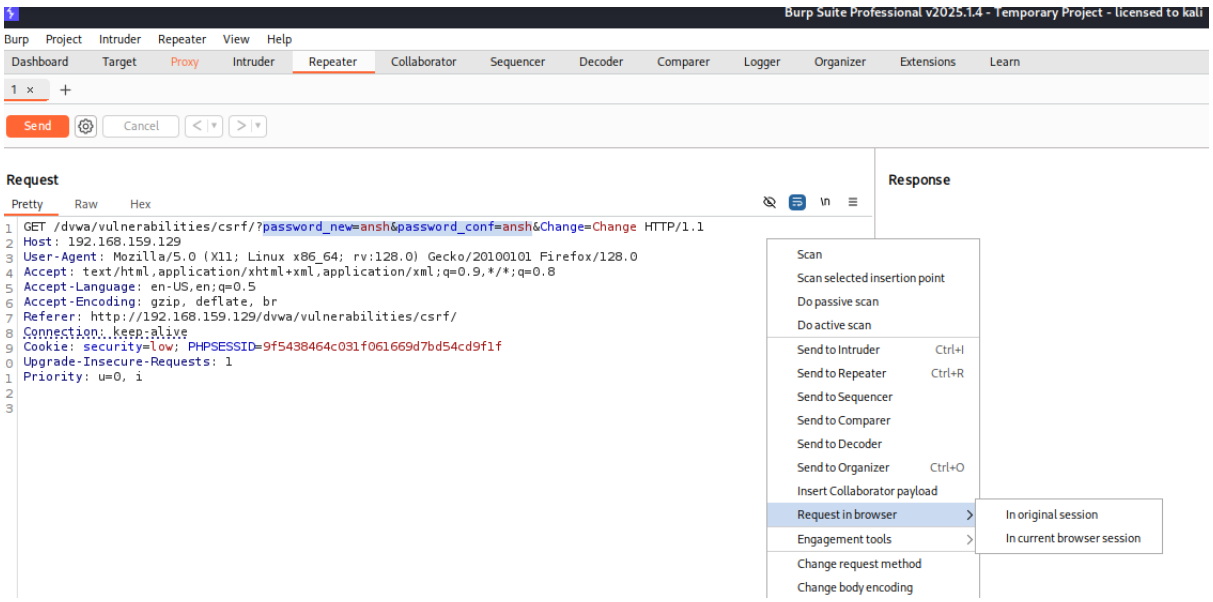
CSRF occurs when an attacker tricks a user into executing unwanted actions on a web application in which they are authenticated, without their knowledge or consent.

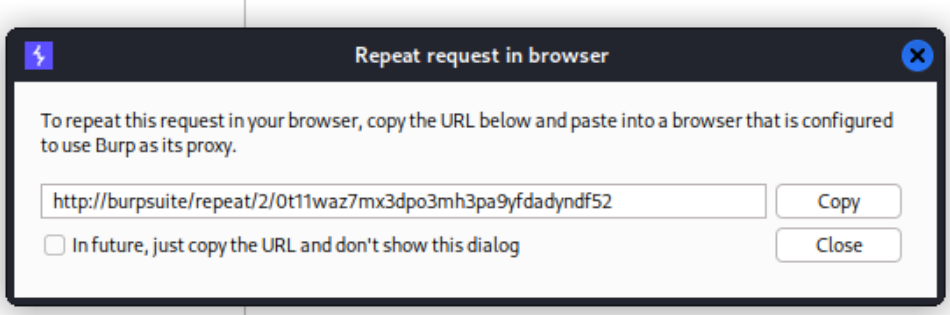
Severity: Medium

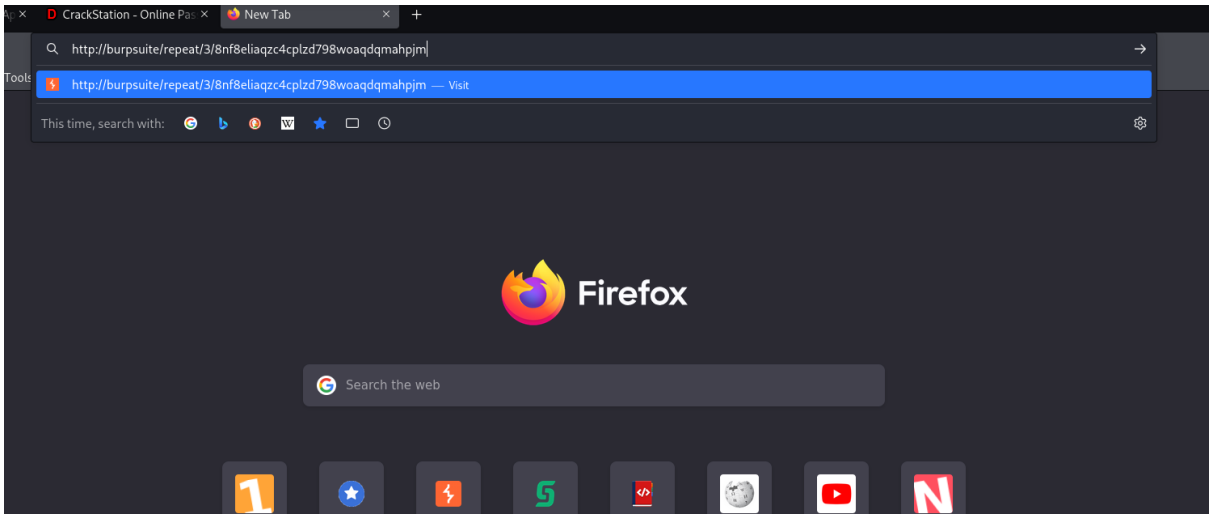
Impact: This vulnerability can lead to unauthorized actions such as changing user settings, initiating transactions, or performing administrative tasks.

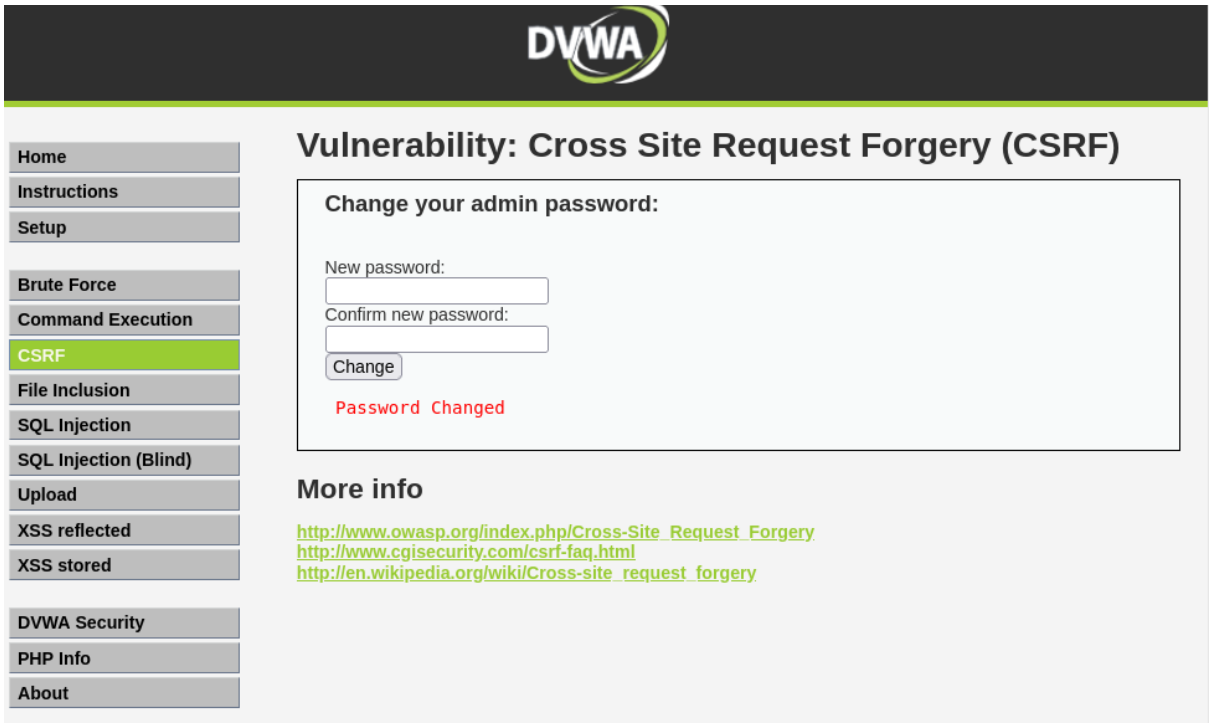
Proof of Concept (PoC):

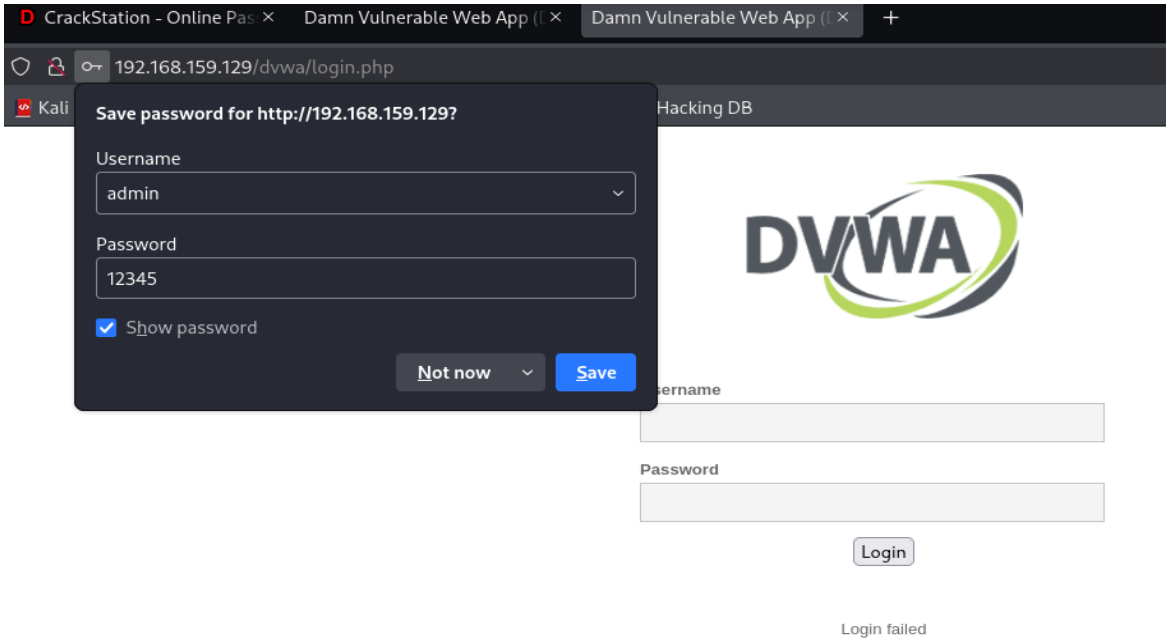


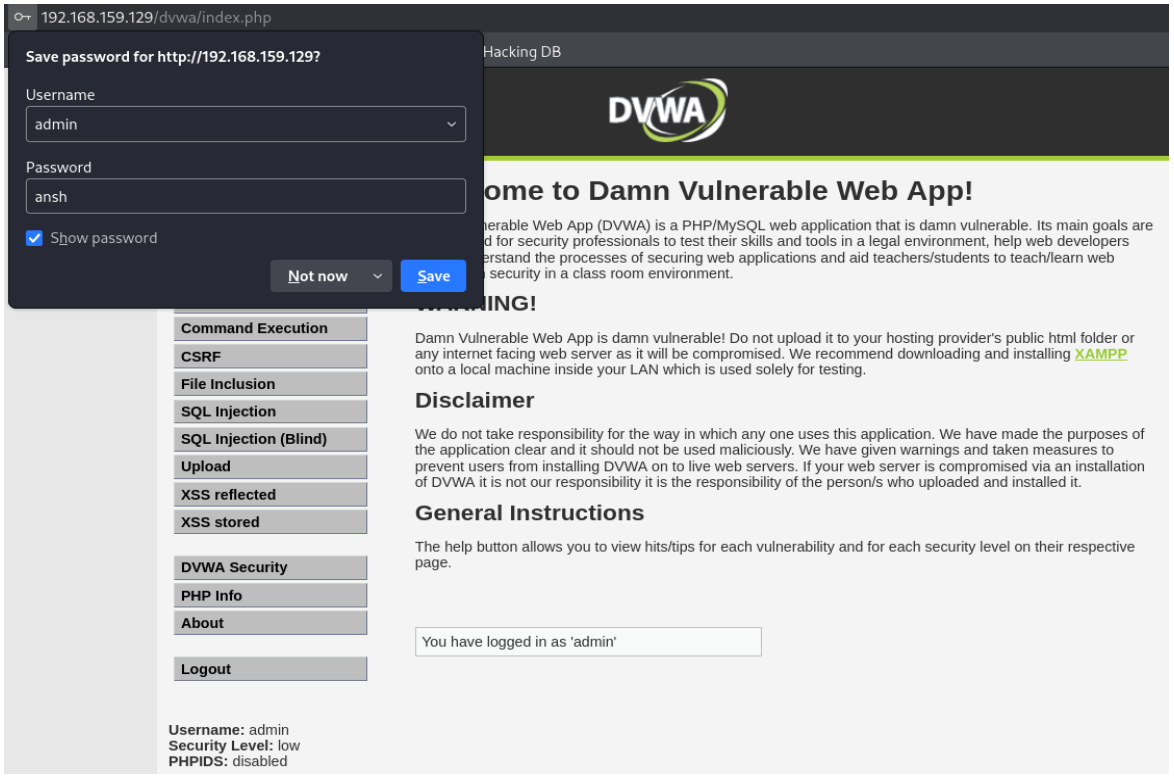



Repeat request in browser
To repeat this request in your browser, copy the URL below and paste into a browser that is configured to use Burp as its proxy.
http://burpsuite/repeat/2/0t11waz7mx3dpo3mh3pa9yfdadyndf52
Copy
In future, just copy the URL and don't show this dialog
Close


CrackStation - Online Pa
New Tab
http://burpsuite/repeat/3/8nf8eliaqzc4cplzd798woaqdqmahpjm
http://burpsuite/repeat/3/8nf8eliaqzc4cplzd798woaqdqmahpjm — Visit
This time, search with:
Firefox
Search the web


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
Vulnerability: Cross Site Request Forgery (CSRF)
Change your admin password:
New password:
Confirm new password:
Change
Password Changed
More info
http://www.owasp.org/index.php/Cross-Site_Request_Forgery
http://www.cgisecurity.com/csrf-faq.html
http://en.wikipedia.org/wiki/Cross-site_request_forgery





Mitigations:

- Add a CSRF token (unpredictable, tied to session) to all state-changing requests and validate it server-side before any action.
- Validate Origin or Referer headers to block requests coming from unauthorized domains.

CHAPTER 4

CONCLUSION

This VAPT project successfully demonstrated how common web vulnerabilities (like SQL Injection, XSS, CSRF, command injection, file upload flaws, and broken authentication) can be identified and exploited in a controlled environment. The exercise bridged the gap between theoretical knowledge and practical skills by exposing real security issues within DVWA, simulating attacker behavior, and documenting actual exploits.

Key Learnings

- **Understanding vulnerabilities:** Gained hands-on experience uncovering OWASP Top 10 issues (e.g., SQLi, XSS, CSRF).
- **Tool proficiency:** Became adept at using Burp Suite and manual exploitation techniques, balancing automated scans with human verification.
- **Severity assessment:** Learned to apply CVSS-based ratings to vulnerabilities, helping prioritize realistic remediation efforts.
- **Reporting best practices:** Documented findings clearly with proof of concept screenshots, payload samples, and remediation guidance—highlighting the importance of clarity and actionability in VAPT reporting.

Future Scope

With additional time or resources, this project could be extended in several impactful ways:

- **Test at higher security levels:** Explore DVWA's Medium, High, and Impossible modes to develop advanced bypass techniques (boolean-based blind SQLi, CSRF tokens bypass, hardened XSS) and supporting proof-of-concept cases.
- **Integrate Continuous VAPT:** Automate scanning with tools like OWASP ZAP or other scanners within a CI/CD pipeline—mimicking DevSecOps practices (e.g., shift left testing, DAST in CI/CD)—for continuous vulnerability detection .
- **Expand to real-world frameworks:** Assess vulnerabilities in modern web technologies—APIs, microservices.
- **Leverage AI/ML tools:** Explore advanced vulnerability detection using AI-enhanced tools or custom scripts

REFERENCES

[1] ken, L. (2023, January 1). DVWA Penetration Testing Report. System Weakness. Retrieved from System Weakness website Journal of Student Research+8System Weakness+8System Weakness+8

[2] PortSwigger. (2024). Burp Suite. Retrieved from https://portswigger.net/burp

[3] Raslan, A. U. U. (2024, August 30). Exploiting and analyzing vulnerabilities in DVWA on Metasploitable2: A comprehensive write up. Medium. Retrieved from System Weakness website

[4] Offensive Security. (n.d.). Kali Linux. Retrieved from https://www.kali.org

[5] VMware, Inc. (n.d.). VMware Workstation. Retrieved from https://www.vmware.com/products/workstation-pro.html